Wireless Innovation Through Software Radios

Dola Saha, Dirk Grunwald, Douglas Sicker
Department of Computer Science
University of Colorado
Boulder, CO 80309-0430 USA
{Dola.Saha, Dirk.Grunwald, Douglas.Sicker}@colorado.edu

ABSTRACT

Advances in networking have been accelerated by the use of abstractions, such as "layering", and the ability to apply those abstractions across multiple communication media. Wireless communication provides the greatest challenge to these clean abstractions because of the lossy communication media. For many networking researchers, wireless communications hardware starts and ends with WiFi, or 802.11 compliant hardware.

However, there has been a recent growth in *software defined radio*, which allows the basic radio medium to be manipulated by programs. This mutable radio layer has allowed researchers to exploit the physical properties of radio communication to overcome some of the challenges of the radio media; in certain cases, researchers have been able to develop mechanisms that are difficult to implement in electrical or optical media. In this paper, we describe the different design variants for software radios, their programming methods and survey some of the more cutting edge uses of those radios.

Categories and Subject Descriptors

C.2.1 [**Communication/Networking and Information Technology**]: Network Architecture and Design; C.3 [**Special-Purpose and Application-Based Systems**]: Reconfigurable hardware; C.2.0 [**Communication/Networking and Information Technology**]: General

General Terms

Design, Performance

Keywords

Software Defined Radio, Orthogonal Frequency Division Multiplexing

1. INTRODUCTION

The use of software defined radio [14] (SDR) and the evolution for use in cognitive radio [16] (CR) has been studied in detail for many years. The systems are becoming common enough, and the programming interfaces capable enough, that many different groups are beginning to explore how software defined radios can be exploited to improve performance or enhance user experience.

There are many underlying motivations behind the advancements in Software Defined Radios (SDRs) [14]. One of the first, and most ambitious, goals was to to solve a persistent problem for the military. Military forces operate in many different regions, each of which has regulatory oversight of spectrum allocation and different communication standards. Developing and deploying radios that could operate across the different ranges of spectrum and implement the different policies needed for deployment in different regions was a difficult task. It would be much easier to have a single radio into which software could be "poured" into the radio to have its behavior conform to the specific locale. Similar motivations exist today - for example, two competing standards for wide-area cellular technology, LTE and WiMAX, both use similar frequencies, waveforms and technologies – although the main difference between the technologies occurs above the physical layer, the physical interfaces are similar enough that it's compelling develop a single radio platform that could handle both standard, including any future variants of the standards.

Standards are another motivation for the the mutable lower layers enabled by software radio. Standards take considerable time to be finalized – a software or mutable platform lets vendors develop and deploy products prior to a fixed standard, and then address any changes once that standard is finalized. Software radios also allow companies to innovate beyond the practice of current standards, and use experimental deployments to assess the value of those extensions. It's also conceivable that a fully software radio might allow vendors to side-step standardization before deployment. For example, many Internet services and development models (*e.g.* REST [9]) are tried out by practical deployments before broad acceptance. Due to the lack of programability, the barrier to innovation in wireless networks is much higher.

All these research efforts constitute a substantial part of the current trends in wireless networks, which require significant communication between physical layer and upper layer protocols with easy reconfigurability of the physical layer hardware platform. In this paper, we describe different ways that researchers envision developing a flexible platform from both the perspectives of hardware and software. We then illustrate the challenges faced by wireless communication and the existing procedures to overcome the challenges Finally we explain how the wireless communication knowledge has been applied in higher layers with reconfigurable radio platforms.

2. PRACTICALITIES OF SDR

Although the concept of software defined radios has been around for a long time [4, 14], the evolving technology in digital signal processing and architecture has made it more practical.

There are four elements needed to enable software radio for high-bandwidth wireless communications: sufficient I/O throughput to transport data from the A/D-D/A devices and the CPU, sufficient digital signal processing throughput, a suitable software development environment and a flexible RF "front end".

2.1 Hardware

The A/D-D/A devices and the radio "front end" translate between the analog RF signals and the digital world of samples. There are many challenges to a purely software implementation of a modern high-speed wireless signal, similar to that of WiFi [10]. The radio interface generates or consumes $\approx$ 160MB/s of data – for a 20Mhz waveform, 40 Msamples per second are needed, and each sample is typically two 16-bit values (representing the phase and the amplitude); until the last two years, this is beyond the ability of most common I/O interfaces. Moreover, the computation needed exceeded the processor throughput of most general-purpose computers in 2007. The challenges for a "narrowband" wireless system are much lower, and Vanu [21] describes such a system.

Many current systems are implemented using *field programmable gate arrays*. For example, Amiri [1] describes a mesh network organization built using custom FPGA boards. Our own group has developed a modular transceiver design using FPGA's and two different design flows - either a traditional "monolithic" implementation [7] and a more modular "system on chip" organization [20]. This latter organization uses FPGA to implement a "sea of components" that are interconnected by a dynamically routed on-chip network, allowing flexibility while producing a specific radio configuration.

A similar approach has been advocated by researchers [15] using the Intel Exoskeleton framework [6] to augment an existing CPU with specific "signal processing components" interconnected by a routable network. The benefit, and limitation, of that approach is that the signal processing blocks are pre-configured, and although the existing blocks can be used in different configurations, blocks cannot be added or changed. This is a limitation because any new signal processing algorithm would need to use the existing blocks; it's a benefit because the chip density and power for such a custom design is typically a 10-fold improvement over FPGA designs. Current designs using FPGA's for signal processing are probably not realistic for low-cost handset designs because of power and circuit density.

An alternative step is to design a CPU that can handle different software radio tasks using distinct instructions optimized for those tasks. Woh. et. al. [23] illustrates the challenges a basic architecture of software defined radio will face when we move to 4G networks. The authors considered the SODA architecture [17] as the basis of their enhancement. This architecture has SIMD support for parallel multiple operations on the same data. The major algorithms that a high-speed network like the WiMax, LTE, and 3GPP standards are comprised of are a) FFT/IFFT, b) STBC, c) V-BLAST and d) LDPC. The FFT/IFFT routines are required for any multicarrier communication, to change the time domain signal to frequency domain and *vice versa*. Since more subcarriers are expected in higher data rate networks, the FFT size can go up to a width of 2048 or more. Space-time coding, STBC, is an encoder/decoder, which is used in MIMO transmission, for encoding two copies of the same data and transmitting it in two time slots. The encoding requires conjugate and negation calculation. Both the encoding and decoding process can be run in parallel and can be computed in a SIMD machine with FFT. V-BLAST is another encoder/decoder technique used for MIMO systems, where spatial multiplexing is obtained by transmitting independent data streams over multiple antennas. It also requires conjugating and negating a block of data. It is an iterative process at the receiver, and consumes computational cycles for implementation. 4G systems are expected to use LDPC codes and also Turbo codes, which is already used in 3G system. LDPC codes are channel encoders, meant for converting the data into a meaningful codeword before transmission to compensate for errors. LDPC also exhibits data level parallelism, which can be implemented in SIMD processor.

The last design alternative used for some software radios is an array of small general "tiled processors" connected by an on-chip network, such as the picoArray [18]. Such designs have been used to build an 802.16 base station.

Each of these design alternatives (general purpose CPU, reconfigurable FPGA, reconfigurable SOC, SIMD processor or tiled array) have advantages and disadvantages. A more fundamental question is *are these capabilities needed?* We believe the answer is "yes". The current trends [13, 12, 19, 8] in software radio innovation require extensive reconfigurability of the hardware and continuous communication from the MAC layer for decision making in signal processing. In legacy hardware, the system would process all the signals similarly and would generate a packet and hand it over to MAC, which then determines what to do with the received packet. At the transmitter side, carrier sensing [19] is done in some subset of subcarriers based on the decision from a statistical model maintained in MAC layer. The physical layer reconfiguration is required depending on the current environment of the network and the received signal (whether it is a collision or not) and requires constant feedback from the MAC layer for signal processing, which cannot be pre-determined or predicted beforehand and programed in an ASIC. This creates a demand for reconfigurable, higher layer controlled physical layer to act to the environment and to the received signal; this requires the capabilities of one of the platforms described above.

2.2 Software

Many on-going projects [13, 12] use the GNURadio [3] framework for signal processing at the receiver side. GNURadio provides a platform of extreme reconfigurability as all the signal processing is done in software code. The software stack is built on the USRP (Universal Software Radio Platform), from Ettus Research, connected to the computer with the USB port, but can be extended to other platforms. The radio front-end up to A/D or D/A conversion is done in hardware, and the rest of the signal processing blocks for both transmitter and receiver are in software. Since a desktop computer is used to compute the signal processing, the processing is serial and takes more time than real demand of the network.

More elaborate (and complete) software environments exist; the SCA (software communication architecture) is a complex framework involving CORBA-based services to configure the radio. An open source version of this software, OSSIE [22], is available, and demonstration of this system also uses the GNU USRP and general purpose CPU's.

The GNURadio framework is mainly appropriate for "non-real time" work because it uses a general-purpose computer. Authors in [19] and [5] use their custom built flexible hardware platform, both of which are FPGA based. One example is the Wireless Open-Access Research (WARP) platform, built on Xilinx Virtex-II Pro FPGA board, where the MAC protocol is written in C and runs on PowerPC cores, and the PHY is implemented in FPGA.

This programming model, which combines C and hardware design languages like Verilog or VHDL, is very challenging for most computer science network researchers. Software, and particularly software accessible by standard networking researchers remains a major challenge for software radio.

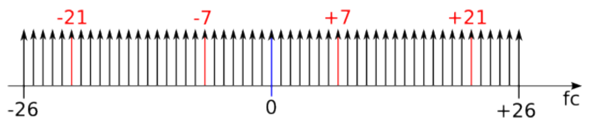


Figure 1: Subcarrier Frequencies in OFDM (Pilots are inserted at Subcarriers -21, -7, +7, +21; Rest are Data Subcarriers)

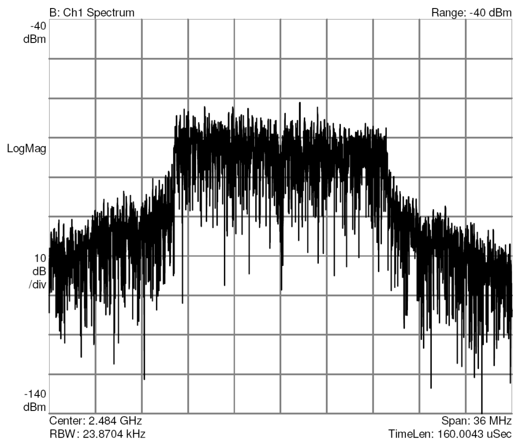


Figure 2: Orthogonal Frequency Division Multiplexed Data Received with Signal Analyzer in 2.4GHz, containing 48 Data Subcarriers and 4 Pilot Subcarriers

3. INNOVATION BY SOFTWARE RADIO

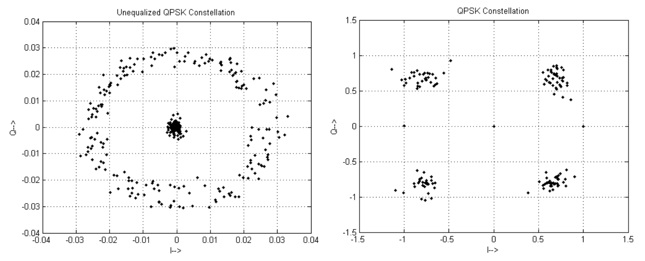
In the next section, we review some of the recent research that requires software defined radio to implement protocols that have shown improvement in the performance of higher layer protocols. We (very briefly) describe the main challenges faced by wireless networks – almost all of these are physical layer issues; understanding how they can be solved by software radios requires a high-level understanding of how radio communication works. We then describe two projects that seek to reduce interference through software processing of the signal. A third project attempts to *reduce interference* by avoiding frequencies being used by other devices. The last project attempts to *exploit* interference to enable rapid group communication and acknowledgements.

3.1 Radio Basics

There are a number of common problems faced by wireless networks, almost all of which are caused by *interference* of the shared communication channel; others are caused by problems in *media access*, such as "hidden terminal" problems. Some of the recent research in using software radios to overcome problems focuses these and other artifacts.

A radio signal is, fundamentally, a sine wave at a specific frequency. That signal needs to be *modulated* to encode different values. Most people are familiar with *amplitude modulation* (AM) and *frequency modulation* (FM). In AM, the amplitude of the signal is changed; but the amplitude is also greatly affected by the environment, which is why AM radios are "noisy". Again, most people are familiar with FM, which modulates the underlying frequency - other techniques such as "frequency shift keying" also use changes in frequency. Less familiar for most people is *phase modulation*, which changes the phase to encode information.

Interference: Fundamentally, two radios transmitting on the



(a) Unequalized QPSK Constellation (b) Equalized QPSK Constellation

Figure 3: Equalization of QPSK

same frequency can be modeled by a sum of time-varying signals, $S_1(t)$ and $S_2(t)$; depending on the strength of one signal *vs.* the other, a receiver may be able to decode one signal or the other. Normally, we separate signals by space, power, time or frequency to insure that the two signals don't interfere to a great extent. For example, FM radio stations and cellular phone networks use frequency and power limits to limit the interference between one radio station and another; 802.11 networks primarily use time (*i.e.* a CSMA/CA MAC protocol) and frequency to limit interference.

Packet Detection: Radios are distributed in space, and the time of reception or start of a packet to any given radio varies by distance. To receive data correctly, most protocols use some method to detect the start of a packet, typically using a *correlator*, that detects a pattern in the transmitted signal. Two common mechanisms [11] are used; one is correlation with stored samples, and the second one is correlation of previously received samples. Correlation determines the degree of similarity between two signals. If the signals are identical, then the correlation coefficient is 1 and if they are completely different, the correlation coefficient is 0. The start of the data symbols are preceded by a known sequence of repetitive preamble symbols. The receiver performs an auto-correlation with a delayed version of the signal. The repetitive nature of the preamble and the delayed auto-correlation function provides us a similarity factor which helps us to identify a valid packet.

Channel Estimation and Equalization: Channel fading distorts the radio signal in both amplitude and phase. In multi-carrier communication, "channel" is defined by as a set of subcarriers, and fading is different across those subcarriers. But, there is a trend in which the subcarriers fade - which can be linear or non-linear. To cope up with this kind of fading in multi-carrier modulation, researchers utilized some subcarriers as *pilots* to capture channel state information. These pilot subcarriers are inserted at regular intervals with a known amplitude and phase. Figure 1 shows four pilot subcarriers inserted in a 52 subcarrier system used in 802.11a/g technology at 2.4GHz. The receiver estimates the channel state information from the pilots and calculates the inter-pilot channel state by using linear or non-linear interpolation methods. These estimates are used to equalize the data carriers in order to reinstate their modulation level required for successful decoding. Figure 2 shows the transmitted OFDM signal captured by a signal analyzer, which contains the four pilot subcarriers and 48 data subcarriers, as shown in figure 1. All the subcarriers overlap in the frequency domain, although the orthogonality (reduced interference) of each of the subcarriers is maintained.

Equalization is an important step in being able to decode the signal. Figure 3 shows the signal before and after equalization of a Quadrature Phase Shift Keying (QPSK) modulated data signal. The

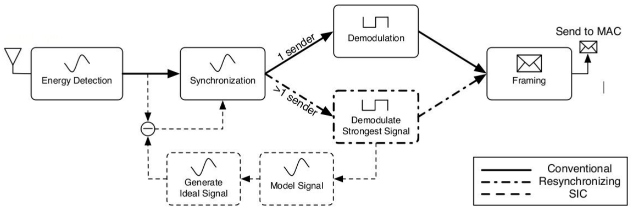


Figure 4: Receiver Architecture for Successive Interference Cancellation [13]

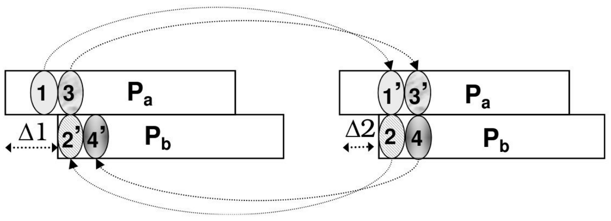


Figure 5: ZigZag Decoding - when two packets collide repeatedly [12]

unequalized data, captured over the air, is distorted to such extent that it is difficult to estimate the values being transmitted. However, a linear equalization results in a distinct "constellation" showing the four different possible states for the QPSK signal. Hence, equalization has to be done before the signal is passed to the demodulation block.

3.2 Interference Cancellation

We now examine how software radio can be used to overcome interference in a signal. Halperin et. al. [13] have implemented an *interference cancellation* algorithm for Zigbee networks, which uses the 2.4GHz band and Direct Sequence Spread Spectrum (DSSS) for its communication. Successive Interference Cancellation (SIC) is a common technique in cellular networks [2], where the scenario is more simple than in the distributed architecture of wireless LANs. A cellular network is controlled by the Base Station(BS), where it constantly communicates with all the mobile devices for clock and frequency synchronization, transmit power, coding rate and spreading codes. However, the chaotic nature of the unlicensed band made it more difficult to address the problem of interference cancellation. Hence, the authors developed a modified version of SIC for unmanaged wireless networks.

SIC is a receiver functionality, that requires a) collision detection, b) decoding the strongest signal, c) modeling the captured signal, d) canceling the strong interferer and e) iterate the process to decode the packet with lower power. Collision detection is done by detecting a sharp change in amplitude of the incoming signal. The authors assume that the signal power of one of the interferers is strong enough that it can be decoded even in the presence of the other signal. Hence, the packet detector module, or more precisely, the correlator block of the receiver resynchronizes with the stronger signal and detects it. Although there are several aspects of channel characteristics, the modeling of the captured signal is done based on a channel model, which only considers frequency offset and ignores others. The received collided signal is a combination of two signals with noise. If $R(t)$ is the received signal at time t, then $R(t) = S_1(t) + S_2(t) + N(t)$. $S_1(t)$ and $S_2(t)$ are the two colliding signals at time t. If we consider that $S_1(t)$ is the stronger signal, then it is decoded by the single packet receiver. After decoding, the original transmitted version of $S_1(t)$ can be regenerated and then channel modeler is used to estimate $S_1(t)$, which results in $S_1'(t)$. This estimated signal is then subtracted from the received signal to decode $S_2(t)$. The block diagram of the SIC receiver used in this research is shown in figure 4.

3.3 ZigZag Decoding

An improvement to Successive Interference Cancellation has been proposed in ZigZag Decoding [12], which also aims to decode two packets when they collide at the receiver due to hidden terminals in an 802.11 network. In the chaotic unlicensed band, hidden terminals are a common problem, causing packets to collide repeatedly at the receiver. When two packets collide more than once at the receiver, ZigZag Decoding receives a *partial* uncollided packet from one collision, decodes it and subtracts it from the signal of the other collision, such that both the packets can be decoded. Figure 5 shows two successive collisions of packets P_a and P_b, which are decoded using the proposed technique. First, the start of each of the collided packets is detected, and $\Delta 1$ and $\Delta 2$ are computed. If $\Delta 1$ is greater than $\Delta 2$, then $(\Delta 1 - \Delta 2)$ is the uncollided part of packet P_a in first collision, which is interfered in the second collision. This part of the samples of packet P_a is called chunk 1, as shown in the figure. Chunk 1 is then subtracted from the signal of the second collision, which retrieves chunk 2 of packet P_b. This is again used to subtract chunk 2 from the first collision, to retrieve chunk 3 of packet P_a. This is done repeatedly till the end of the packets or until both the packets are decoded. The size of each chunk depends on the size of $(\Delta 1 - \Delta 2)$, which is called the bootstrapping chunk. The authors also show that this procedure can be theoretically extended to collisions of more than two packets.

This technique requires following the extra steps at the receiver: a) detection of the start of each of the packets under collision, b) detection of the end of packet, c) channel estimation, and d) detection of matching collisions. The detection of the start of packet is done by cross-correlation with a known sample. Once a packet is detected, a sliding window of correlation is used to detect the start of next packet. When samples match, the correlation has a high peak denoting the start of the packet. Although, if the first signal is strong enough, there is a possibility of not getting a peak above the threshold in the cross-correlator block for detection of the second packet. If the preamble and signal symbol is detected and decoded correctly, length of the packet is retrieved from the signal symbol and is used to decode the packets in a loop until the end. Channel estimation is part of any receiver design, where the equalization block is required to regenerate the transmitted signal. In figure 5, we notice that chunks are named as 1 in the first collision and $1'$ in the second collision, which is due to the fact that channel conditions vary and time-dependent signals are not identical. ZigZag first decodes chunk 1 with the demodulator known from the signal symbol, and encodes chunk back, which generates the original transmitted signal devoid of any channel distortion that modifies the signal. This signal is then passed through the channel estimator to generate a similar, but not necessarily identical copy of the chunk 1 or $1'$, which is then subtracted from the signal of second collision. To detect a matching collision, correlation is done with the samples of $(\Delta 1 - \Delta 2)$ and the recently collided signals at the receiver. The receiver stores signals of all recent collisions.

3.4 SWIFT

The use and demand of unlicensed bands has been increasing to satisfy the needs of narrowband users like 802.11b/g or Zigbee. However, wideband technologies can be used to transmit media rich applications at a higher data rate if they can successfully coexist with other narrowband technologies and form a network of their own. Rahul et. al. [19] presents the design and implementation of such a narrowband-friendly wideband network that tries to coexist with other narrowband technologies. The technology is termed *SWIFT*, a Split Wideband Interferer Friendly Technology. SWIFT introduces an Adaptive Sensing Algorithm, which inverts the common use of sensing the power of the signal at each frequency. Instead, the algorithm senses the channel for the reaction of the narrowband user in the presence of the wideband transmission. The adaptive sensing performs power measurements in subcarriers to compute four metrics, to determine how the narrowband network reacts to the wideband transmission. The four metrics are a) Inter-transmission time, b) Transmission duration, c) Average narrowband power and d) Probability of transmission immediately after SWIFT. The first metric is to determine the backoff nature of the 802.11 device. The second metric is to determine whether the data rate falls back due to interference, which often does not happen if the data rate is fixed. The third metric is to determine the presence of multiple narrowband devices. The fourth one considers that 802.11 will carrier-sense and then transmit after wideband transmission, if it can sense. With these metrics, adaptive sensing is done, based on the algorithm as shown in figure 6. Packet detection is done by filtering out the frequencies reported in the adaptive sensing mechanism. However, this requires a communication between a transmitter and receiver pair to agree upon a common subset of subcarriers for communication.

3.5 Concurrent Acknowledgement

In on-going work, we [8] have shown how to utilize orthogonal subcarriers in OFDM as a mode of *simultaneous transmission* by multiple nodes. This mechanism is aptly used for a reliable broadcast or multicast protocol, where the acknowledgements are transmitted simultaneously by the clients who received the packet in a predefined subcarrier. This can also be extrapolated to include some decision mechanism at the clients to "vote" on the transmitted message. For the transmission of acknowledgements, each node only transmits a short tone of duration $8\mu s$ in a pre-negotiated subcarrier. The received signal at the receiver is a cumulative signal from all the nodes. As the subcarriers are mutually orthogonal to each other, the frequencies in which tones are transmitted are distinctly visible after FFT. The tones just carry 'yes/no' information based on either having the nodes transmit or not. Hence, these tones do not need to be decoded; only checking the amplitude of each subcarrier suffice the decision making process of which node has transmitted the tone. The idea has been modeled in MATLAB and then implemented in an FPGA-based reconfigurable hardware [10, 7]. Figure 7 illustrates how radios can simultaneously "vote" using orthogonal (non-overlapping) communication frequencies. This "waterfall plot" shows energy at different frequencies (horizontal axis) over time (vertical axis). The wideband energy at the top of the figure is a broadcast packet asking nodes to respond if some condition is met; the two bands near the bottom of the plot are responses from two nodes.

This purely physical layer communication mechanism can be used to dramatically improve various MAC and network functions. In unpublished work, we've shown how such concurrent acknowledgments can be used to greatly improve the throughput of an 802.11-like protocol by reducing or eliminating contention while maintaining fair scheduling. It's also possible to use aspects of the physical layer information on "time of arrival" to estimate velocity and distance in vehicular networks.

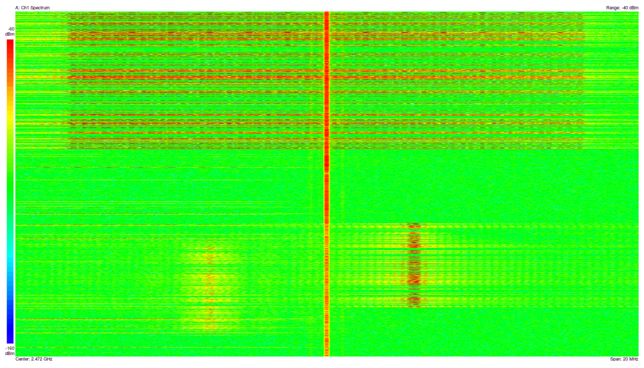


Figure 7: Waterfall Plot of Concurrent Acknowledgement with 3 Radios showing the end of transmission of broadcast packet using full spectrum and 2 Concurrent Acknowledgements from 2 Clients, each using one Subcarrier [8]

4. CONCLUSION

Each of the preceding innovating uses of wireless networks has used a combination of *physical layer* characteristics, often in conjunction with *MAC layer* capabilities, to improve some aspect of wireless communication, such as reducing interference, enabling better frequency utilization or reducing the time needed for signaling and coordination. None of these innovations would be possible without a *fully software programmable* radio, where the physical layer and MAC layer are under direct programmer's control.

The hidden cost of these projects is the complexity of the development environment and the time needed to implement them. This is in part because of the tension between the needed capabilities of the hardware and the (rather arcane) programming tools needed to fully exploit that hardware. Based on our work, we believe that software radio needs a broader focus on the software to simplify development and enhance innovation.

In this paper, we have seen that a reconfigurable physical layer, which can efficiently and seamlessly interact with higher layers, will be a preferred choice for researchers. We reviewed four innovation that utilize flexible physical layer design in higher layers for the overall improvement of system performance. Instead of treating the MAC and physical layer as separate entities, software developers need to be able to treat these layers as the mutable components in order to continue the evolution of wireless networks.

5. REFERENCES

[1] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. R. Cavallaro, and A. Sabharwal. WARP, A Unified Wireless Network Testbed for Education and Research. In *Proceedings of IEEE MSE*, 2007.

[2] J. Andrews. Interference cancellation for cellular systems: a contemporary overview. *Wireless Communications, IEEE*, 12(2):19–29, April 2005.

[3] E. Blossom. GNU Radio as an Experimental Platform: Current Capabilities and Future Directions. In *WINTECH*, pages 1–2, 2007.

[4] V. G. Bose. The impact of software radio on wireless

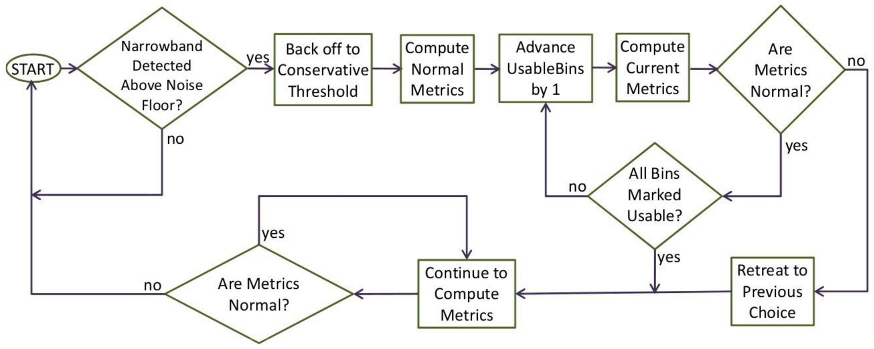


Figure 6: Adaptive Sensing Algorithm of SWIFT [19]

networking. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(1):30–37, 1999.

[5] J. Camp and E. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. In *MOBICOM '08: Proceedings of the ACM MOBICOM 2008 conference*, New York, NY, USA, 2008. ACM.

[6] G. Chinya, J. Collins, M. Girkar, H. Jiang, G. Lueh, L. Pearce, X. Tian, H. Wang, P. Wang, and S. Yakoushkin. Accelerator exoskeleton. *Intel Technology Journal*, Aug 2007. `http://www.intel.com/technology/itj/2007/v11i3/2-exoskeleton/1-abstract.htm`.

[7] A. Dutta, J. Fifield, G. Schelle, D. Grunwald, and D. Sicker. An intelligent physical layer for cognitive radio networks. In *WICON 2008: Proceedings of the Fourth International Wireless Internet Conference (WICON 2008)*, New York, NY, USA, 2008. ACM.

[8] A. Dutta, D. Saha, D. Grunwald, and D. Sicker. A concurrent acknowledgement scheme for broadcast messages in wireless networks. Technical report, University of Colorado at Boulder, September 2008.

[9] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, Department of Information and Computer Science, University of California Irvine, 2000.

[10] J. Fifield, P. Kasemir, D. Grunwald, and D. Sicker. Experiences with a Platform for Frequency-Agile Techniques. In *DYSPAN 2007*, 2007.

[11] A. Fort, J.-W. Weijers, V. Derudder, W. Eberle, and A. Bourdoux. A performance and complexity comparison of auto-correlation and cross-correlation for ofdm burst synchronization. *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 2:II–341–4 vol.2, April 2003.

[12] S. Gollakota and D. Katabi. Zigzag Decoding: Combating Hidden Terminals in Wireless Networks. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 159–170, New York, NY, USA, 2008. ACM.

[13] D. Halperin, T. Anderson, and D. Wetherall. Taking the Sting out of Carrier Sense: Interference Cancellation for Wireless LANs. In *MOBICOM '08: Proceedings of the ACM MOBICOM 2008 conference*, New York, NY, USA, 2008. ACM.

[14] J. M. III. *Cognitive Radio - An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, Royal Institute of Technology (KTH), 2000.

[15] D. A. Ilitzky, J. D. Hoffman, A. Chun, and B. P. Esparza. Architecture of the scalable communications core's network on chip. *IEEE Micro*, 27(5):62–74, 2007.

[16] F. K. Jondral. Software-defined radiobasics and evolution to cognitive radio. *EURASIP Journal on Wireless Communications and Networking*, 2005(3):275–283, 2005.

[17] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. Soda: A low-power architecture for software radio. In *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, pages 89–101, Washington, DC, USA, 2006. IEEE Computer Society.

[18] G. Panesar, D. Towner, A. Duller, A. Gray, and W. Robbins. Deterministic parallel processing. *Int. J. Parallel Program.*, 34(4):323–341, 2006.

[19] H. Rahul, N. Kushman, D. Katabi, C. Sodini, and F. Edalat. Learning to Share: Narrowband-Friendly Wideband Networks. In *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 147–158, New York, NY, USA, 2008. ACM.

[20] G. Schelle, J. Fifield, and D. Griinwald. A software defined radio application utilizing modern fpgas and noc interconnects. *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pages 177–182, Aug. 2007.

[21] D. L. Tennenhouse and V. G. Bose. The spectrumware approach to wireless signal processing. *Wirel. Netw.*, 2(1):1–12, 1996.

[22] `Wireless@VirginiaTech.edu`. Ossie: Open source sca implementation. `http://ossie.wireless.vt.edu/`, 2008.

[23] M. Woh, S. Seo, H. Lee, Y. Lin, S. Mahlke, C. Chakrabarti, and K. Flautner. The Next Generation Challenge for Software Defined Radio. In *SAMOS*, 2007.