A Real-Time and Protocol-Aware Reactive Jamming Framework Built on Software-Defined Radios

Danh Nguyen, Cem Sahin, Boris Shishkin, Nagarajan Kandasamy, and Kapil R. Dandekar
Drexel Wireless System Laboratory
Drexel University, 3141 Chestnut St., Philadelphia, PA 19104
{dnguyen, cs486, bs44, kandasamy, dandekar}@drexel.edu

ABSTRACT

This paper develops a software-defined radio (SDR) framework for real-time reactive adversarial jamming in wireless networks. The system consists of detection and RF response infrastructure, implemented in the FPGA of a USRP N210 and designed to function with the open source GNU Radio SDR library. The framework can be used to implement a fast turnaround reactive jamming system capable of timely RF response within *80ns* of signal detection. Our framework also allows for full control and feedback from the FPGA hardware to the GNU Radio-based cognitive radio backend, making it applicable to a wide range of preamble-based wireless communication schemes. This paper presents the capabilities, design, and experimental evaluation of this framework. Using this platform, we demonstrate real-time reactive jamming capabilities in both WiFi (802.11g) and mobile WiMAX (802.16e) networks and quantify jamming performances by measuring the network throughput using the *iperf* software tool. The results indicate that our system works reliably in real time as a reactive jammer and can be used for practical assessments of modern jamming and secure communication techniques.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security, Experimentation, Performance

Keywords

Denial of Service; Reactive Jamming; Software Radio

1. INTRODUCTION

Software-defined radios have been instrumental in multiple areas of research in wireless communications, especially those requiring real-time experimentation. SDRs allow highly customizable hardware and software designs to be used as testbeds for prototyping new communication techniques without the need for proprietary ASIC hardware. Due to shorter development time and the flexibility they afford across the hardware-software stack, SDRs are an attractive choice for proof-of-concept experimentation and performance analysis within the wireless research community.

Recent research in wireless communications has strongly emphasized securing the physical layer against external threats. One such threat is the Denial-of-Service (DoS) attack, wherein the adversary transmits interfering signals, i.e. jamming signals, to make the network unavailable to legitimate users. In its most basic form, a DoS attack can just be a continuous inband jamming signal with sufficient power to corrupt all transmitted packets. A continuous jammer, though simple to implement, suffers from two disadvantages: high power requirement and high probability of detection. On the other hand, reactive jammers are more efficient due to their ability to sense the wireless medium and jam packets that are already in the air [3]. By jamming wireless packets reactively at critical moments, adversaries can significantly reduce network throughput using little energy while minimizing the chances of being detected. Nevertheless, reactive jammers have not been considered a serious threat in practice, mainly due to the implementation challenges in meeting strict real-time constraints for detecting and reacting to in-flight packets of high-speed wireless networks [3].

This paper develops an SDR platform capable of very fast RF signal detection and response, to be used in wireless security research. The main focus of discussion is a prototype of a real-time, multi-standard reactive jamming system for preamble-based communication networks, implemented using GNU Radio [1] and the USRP N210 SDR platform from Ettus Research. We show that reactive jammers can be realized using readily available, commercial off-the-shelf SDR hardware, but nonetheless can achieve the necessary performance to reliably and selectively jam in-flight packets of WiFi (802.11 a/b/g) and mobile WiMax (802.16e) networks. We anticipate that the developed platform will be quite useful in wireless security research, both to realize a range of sophisticated DoS attacks as well as to prototype several classes of jamming-based secure communication schemes.

There is considerable prior research analyzing the threat posed by adversarial jamming to several wireless protocols [3, 4, 13, 15]. The problem of reactive jamming, in particular, has been studied from both the viewpoint of a jammer to devise optimal jamming strategies [7, 10], and from the viewpoint of a wireless network to achieve jamming-resilient communications [8, 9, 12]. Recently, the possibility of using self-jamming and cooperative jamming as a way to create secure wireless networks has also been considered. Gollakota and Katabi [5,6] develop a data secrecy scheme called iJam wherein randomized self-jamming signals are used to deny potential eavesdroppers access to the raw signal data. Similarly, Shen et al. [11] develop a method to jam the wireless channel continuously while properly controlling the jamming signals with secret keys such that these signals interfere in an unpredictable fashion with unauthorized devices but are recoverable by authorized ones equipped with the secret keys.

While a majority of the above-mentioned research focuses on theoretical analysis and simulations, researchers still lack a practical way to deploy experimental protocols and evaluate their performances in typical high-speed wireless environments. The difficulties of achieving synchronization with real-time signals and timely RF responses are outlined and echoed throughout many publications. For example, the iJam protocol was experimentally demonstrated using USRP radios; but the transmitter must purposely introduce dummy paddings at the end of the PHY header, before the useful data, to account for the decoding and jamming response delays at the receiver. During our literature review, only a single study, by Wilhelm et al. [14], was found to perform reactive jamming using SDRs on standard-compliant networks in real time; the authors demonstrate a hardware implementation of reactive jammers capable of operating in low-rate, Zigbee-based 802.15.4 networks.

The primary contribution of our paper is a reactive jamming platform with significantly faster RF response time as well as additional degrees of freedom for performing live experiments with a variety of high-speed wireless standards. The paper is organized as follows: Section 2 discusses design and implementation details of the reactive jamming framework. We characterize the platform's detection performance in WiFi 802.11g networks in Section 3. Validation results for both WiFi and WiMAX networks are presented in Sections 4 and 5, respectively, and we conclude the paper in Section 6.

2. SYSTEM ARCHITECTURE

2.1 SDR Platform Overview

The USRP N210 SDR hardware and GNU Radio framework provide a low-cost, open-source starting point for development. While other SDR platforms exist with potentially faster over-the-air response time (e.g., WARP [2]), we choose the USRP for our implementation due to its existing integration with several signal intelligence libraries implemented in GNU Radio and the ease of composing jamming waveforms on the fly. The USRP N210 is capable of full-duplex transmission and supports several front-ends, including the SBX radio daughterboard that we use in this design. SBX is an agile transceiver board that provides up to 40 MHz of instantaneous RF bandwidth and tunable center frequency between 400 MHz and 4 GHz. This flexibility allows us to conduct reactive jamming experiments with a variety of high-speed wireless standards. To minimize switching time between RX and TX operations during jamming, we initialize both TX and RX chains simultaneously in the host application at start-up. All time-sensitive functions, including packet detection, trigger filtering, and transmit-to-jam operations are implemented in the FPGA hardware, with

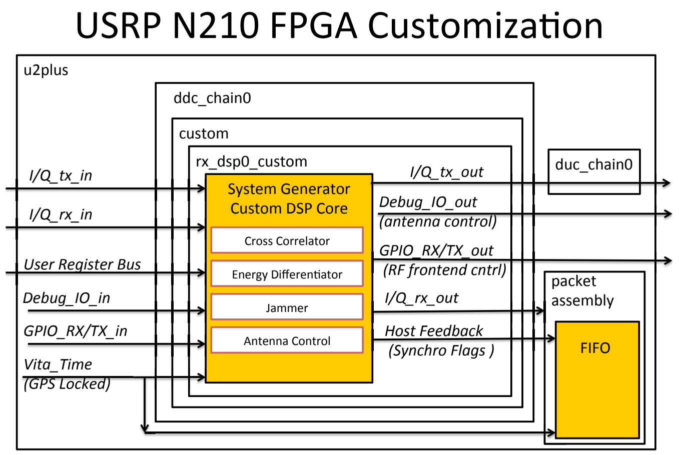


Figure 1: High level overview of the custom IP implementation using the USRP N210 FPGA.

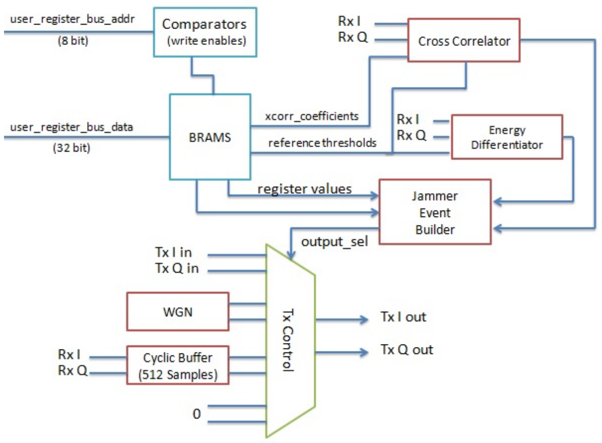


Figure 2: The hardware block diagram of the reactive jammer.

only a few high-level controls provided to host. This implementation effectively bypasses host-side operations and even the soft processor on the USRP during signal processing. Our custom DSP core takes complete control of the transmit data path and produce jamming waveforms based on several application presets. A wide variety of detection settings and jamming response parameters, such as jammer uptime, delay, and waveforms, are dynamically accessible and can be changed on the fly from the host application.

To increase the platform flexibility, we target a hardware system that works with the highest sampling rate available on the USRP platform. As such, the hardware design is built to work with a USRP baseband sampling rate of 25 MSPS, and a hardware clock of 100 MHz.

2.2 Custom Hardware Implementation

The Universal Hardware Driver (UHD) for USRP products allows for the customization of DSP operations at multiple critical locations in the digital downconversion chain (DDC). The core of our design, a hardware block that acts as a custom packet detector and a jamming controller, is nested within the DDC chain and inside the custom DSP module wrapper provided by UHD, as shown in Fig. 1.

Figure 2 shows the high-level architecture of our custom DSP core, consisting of four main functional blocks: a cross-correlator for matched filtering, an energy differentiator for energy rise and fall detections, a jamming event builder, and a transmit controller. Details on each of these functional blocks are covered in subsequent sections. In addition to these main blocks, the core also has a number of smaller logic blocks responsible for internal timing synchronization and control of external I/Os. I and Q samples of the received RF signal, after down-conversion, decimation, and filtering, are passed through this custom core to carry out user-defined operations, including signal detection and jamming triggers. We implement our custom DSP core using Xilinx System Generator (version 14.2), a plugin for Matlab Simulink that allows HDL generation directly f rom block-level diagrams.

The host-side control path for the core is enabled through the user register bus in the UHD design. This user bus contains a 32-bit data bus and an 8-bit address bus, together providing up to 255 programmable 32-bit registers in the user's custom DSP core. The `gr-uhd` component of GNU Radio provides the necessary APIs to interact with this user register bus and program the custom user registers to control our DSP core. Our current design makes use of 24 of these user registers to enable run-time updates of cross-correlator coefficients, detection thresholds, jammer settings, and antenna control signals from host applications. Figure 2 shows the internal architecture of our custom DSP core, together with the primary I/O signals routed from higher levels of the UHD hardware implementation.

2.3 Hardware Signal Detection

We make several design choices in designing our signal detection system to ensure successful and accurate jamming in real time, while minimizing the occurences of false reactions and retaining system flexibilty. First, we move signal detection processing from host onto the USRP N210's FPGA. This allows for high-speed detector designs and deterministic timing in operations. Secondly, in our custom DSP core, a signal cross-correlator and an energy differentiator operate in parallel to realize fast and accurate real-time signal detection. The cross-correlator performs template-based detection and enables the platform to react to only packets of a single wireless standard. The energy detector performs coarse-grained detection to detect any wireless activity on a particular band of interest. The signal detection hardware also provides a control path for customization of detection logic at runtime.

Signal cross-correlator: For synchronization and fine-grained signal detection, we extract and make use of the cross-correlation DSP core from Rice University WARP's OFDM Reference Design version 15 [2]. This referenced hardware core implements a 64-sample weighted phase correlator, with 90° phase resolution, using the sign bits from each pair of incoming I and Q samples. The structure of our version of the core, with added custom logic, is shown in Fig. 3. In our design, incoming baseband samples are correlated against a template of 64 3-bit signed cross-correlation coefficients for both I and Q signals. These coefficients are generated offline on the host based on knowledge of the wireless standards' preambles or inferred from the low-entropy portions of the samples of incoming signals. The result of this process is a confidence-weighted phase correlator output that is then compared against a user-selected threshold

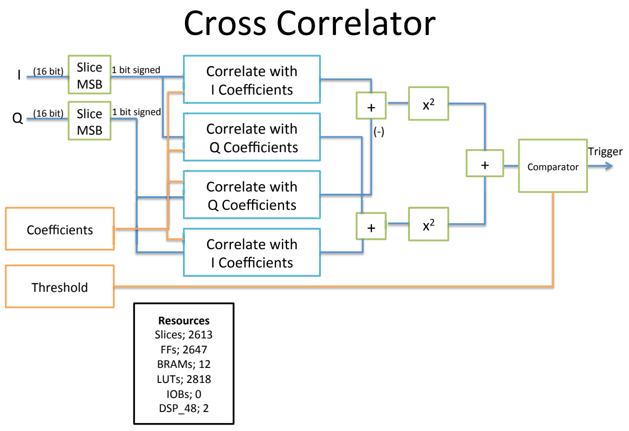


Figure 3: Block diagram of the cross-correlator block on the FPGA.

for a detection decision. In addition, we also modify the referenced cross-correlation core to enable run-time loading of user-defined correlation coefficients from host-side applications through the user register bus provided by UHD. This allows full customization of signal detection as long as the host application knows the preset preamble values or is able to detect some low-entropy portions of the received signal.

Energy Differentiator: Our secondary detection method is an energy differentiator, shown in Fig. 4, which continuously compares the energy level of incoming samples against the recent past to detect an energy rise or fall. In essence, this hardware block keeps a running sum of N recent energy readings, where N is the desired length of the differentiator (32 samples in our experimental implementation). At the nth instant, an energy reading $x[n]$ is computed from the incoming pair of I and Q values. The energy sum $y[n]$ is then updated according to the relationship:

$$y[n] = y[n-1] + x[n] - x[n-N], \text{ for } n \geq N$$

The output of the energy sum calculator is compared to its own previous values after scaling by user-defined thresholds either for energy high or energy low detection. Users can set detection for any energy level change between 3dB and 30dB, and for both positive (increasing) and negative (decreasing) energy changes. This energy differentiation provides the channel occupancy status if no cross-correlation coefficients are available.

2.4 Real Time System Response

In our jamming platform, reactive jamming capability is achieved through triggering jamming operations immmediately following detection events. In our initial implementation, a three-stage hardware state machine allows the user to select up to three trigger event combinations, all of which must occur within a user-assigned time interval. Once triggered, jamming operations will take place using one of three user-selectable waveforms: *(i)* a pseudorandom 25MHz White Gaussian Noise (WGN) signal, *(ii)* a repetitive replay of up to 512 most recently received samples, or *(iii)* the waveform currently being streamed to the transmit buffer from host.

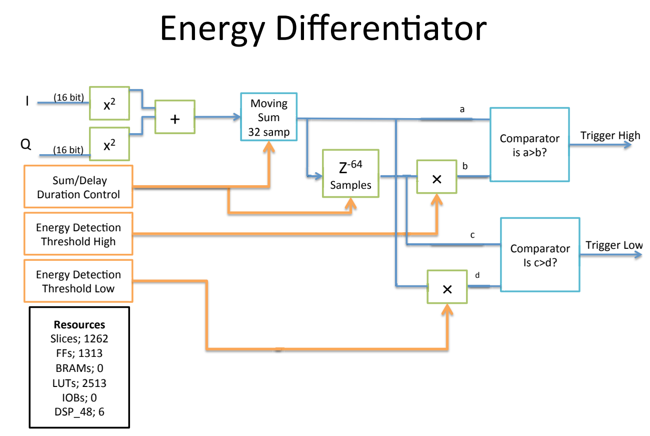


Figure 4: The energy differentiator block diagram.

The duration of jamming is also customizable and can range from 1 sample time ($40ns$) to 2^{32} sample time (about $40s$). A user-defined delay option between detection triggers and active jamming is also provided to enable jamming of specific locations in the packets.

Since all detection and reaction functions are implemented in FPGA hardware, the turnaround time between detection triggering and jamming response is extremely short. An RF jamming response can be initiated within 1 clock cycle of detection trigger, with approximately seven more cycles required to populate the digital up-conversion chain (DUC) with jamming waveforms. With the 100 MHz hardware clock of the USRP N210 platform, our platform can detect and jam over-the-air packets within *80ns* of signal detection.

2.5 Host Side Interface

We implement a Python-based custom GUI to configure our jammer operations on the fly, using GNU Radio Companion with additional custom code. This GUI acts as a reactive jamming event builder, where users can specifically control detection types and desired jamming reactions during run time. The user inputs are passed directly to the UHD driver stack, which uses pre-defined functions to set all RF as well as our custom DSP core operations. The GUI application is a useful tool for demonstration purposes, and can be easily modified to provide an interface for more powerful host side processing applications, thereby enabling complete, autonomous jamming operations.

3. DETECTION PERFORMANCE

3.1 Jamming Timelines

To get an impression of the platform's reactive jamming capabilities, we estimate the timelines of various detection and jamming operations based on their latency in terms of hardware cycles and show our analysis in Fig. 5. Here we denote the minimum time for the system to detect an energy high (or low), corresponding to an active transmission, as T_{en_det}, the time for cross-correlation detection as T_{xcorr_det}, the time to schedule and initialize the TX pipeline for jamming as T_{init}, and the jamming duration as T_{jam}. The system response time, denoted as T_{resp}, is the combined time for both detection and jamming triggers. We make the following observations:

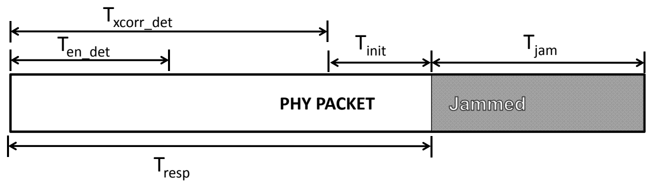


Figure 5: Reactive jamming timelines.

- An energy high (or low) detection takes at most 32 baseband samples, or 128 clock cycles, to trigger. This duration can be shorter if the threshold is set low (at the cost of higher false alarm rates). Since the hardware clock is 100 MHz, we have $T_{en_det} < 1.28\mu$s.
- The cross-correlator hardware performs a 64-sample phase correlator. With a good preamble design, it takes exactly 64 samples from the start of transmission to trigger a cross-correlation detection. At a digital sampling rate of 25 MSPS, this means $T_{xcorr_det} = 2.56\mu$s.
- Once a detection triggers, it takes about 8 clock cycles to initialize the transmit chain and start jamming, so $T_{init} \approx 80$ns. The system response time is therefore less than 1.36μs if using energy detection, and 2.64μs using cross-correlation detection.
- The jamming duration (T_{jam}) is selectable between $40ns$ and $40s$ by the users. In general, a short but sufficient jamming burst is desired. Jamming can also be inititalized after a custom delay to target specific portions of the packet. This type of "surgical" jamming is highly destructive due to its ability to target critical information contained in a wireless PHY packet, such as channel estimation [13].

In comparison, each 802.11g WiFi packet contains 10 short preambles (8μs duration), 2 long preambles (8μs duration), PHY parameters (4μs) and a variable length Physical Layer Service Data Unit (PSDU). With a system response time of at most 2.56μs, an 802.11g packet can be jammed before the first OFDM data symbol is received.

3.2 Signal Detection

We characterize the signal detection performance of the platform in a WiFi transmission scheme. As mentioned earlier, the signal detection subsystem consists of a cross-correlator and a differential energy detector for two different detection purposes. The cross-correlator is useful for fine-grained detection of signals with known preambles from a particular wireless standard. On the other hand, the differential energy detector is useful when no signal preamble is known before hand, and the user's goal is to detect any kind of RF signals on a specific band of interest.

In one experiment, the cross-correlator is set up to detect two types of WiFi preambles: the short preambles consisting of 16 samples in length, and the long preambles consisting of 64 samples in length. To test the robustness of this cross-correlator, we use a second USRP N210 as the transmitter and generate two types of frames for testing: complete WiFi frames with 10 short preambles, 2 long preambles, the SIGNAL symbol, and the payload, as well as pseudo-frames with only a single short or long preamble. Our characterization is performed in a wired link to isolate environmental effects and provide independently measured SNR values at the receiver.

To get the characteristic false alarm rates for a particular correlation threshold, we terminate the receiver with a 50Ω terminator and count the number of false triggers that occur in 30 minutes. For probability of detection, we generate and send 10000 WiFi frames (or pseudo frames), at 130 frames per second, and count the number of detections. Figure 6 shows the detection results of long WiFi preambles under two false alarm rates, 0.083 and 0.52 triggers/s. We see that using a lower correlation detection threshold, i.e., aiming for a lower false alarm rate, generally decreases the probability of detection. For detection of a single long WiFi preamble, the detection rate increases with higher SNR and is slightly above 50% for SNR over 5 dB, which is rather low for a 64-sample correlator. The efficiency of the cross-correlator depends on a number of parameters, including the sampling rate mismatch between the correlator and the RF signal, the dynamic range characteristics of the signal being correlated, and the quantization of both the phase and amplitude of the correlation coefficients. In this case, the sampling rate mismatch between the transmitter and receiver causes a highly suboptimal operating condition for the cross-correlator. On the transmit side, the WiFi long preamble is generated assuming a sampling rate of 20 MSPS according to the 802.11g standard (64 samples within a period of 3.2μs FFT integration time). On the receive side, the correlation window also consists of 64 samples, but at a digital sampling rate of 25 MSPS as dictated by the UHD hardware design. As a result, an orthogonal code that is 3.2μs long is being correlated across its first 2.56μs, yielding significantly low detection performance. Increasing the correlation size above 64 samples will undoubtedly improve the single-preamble detection performance, but will also give rise to higher resource utilization, lower speed, and potential timing issues.

Figure 6 also shows that the probability of detection becomes better if multiple copies of the same preamble are transmitted, as in the case of full WiFi frames. Since two long preambles are transmitted in each WiFi frame, the cross-correlator yields significantly higher detection rates at over 75% for SNR above 5 dB.

For comparison, we conduct the same test using full WiFi frames but set the cross-correlator to detect only WiFi short preambles. Each short preamble consists of 16 samples at 20 MSPS, repeated for 10 times in a WiFi frame. The duration of the orthogonal code here is 0.8μs with 10 cyclic repetitions, totaling 8μs of short preamble time. The detection results, shown in Fig. 7, show that the cross-correlator is able to trigger on over 90% of the preambles at -3 dB SNR, and over 99% of the preambles at SNRs above 3 dB, with a constant false alarm rate of 0.059 detection per second.

Using the same methodology, we characterize the energy differentiator detection performance by sending complete WiFi frames at 130 frames per second, for a total of 10,000 frames. The energy detection threshold is set to 10 dB, yielding a measured false alarm rate of 0 detection per second. Figure 8 shows the energy high detection performance when the received SNR is increased gradually. For SNRs below -3 dB, the signal level is below the noise floor, and thus no

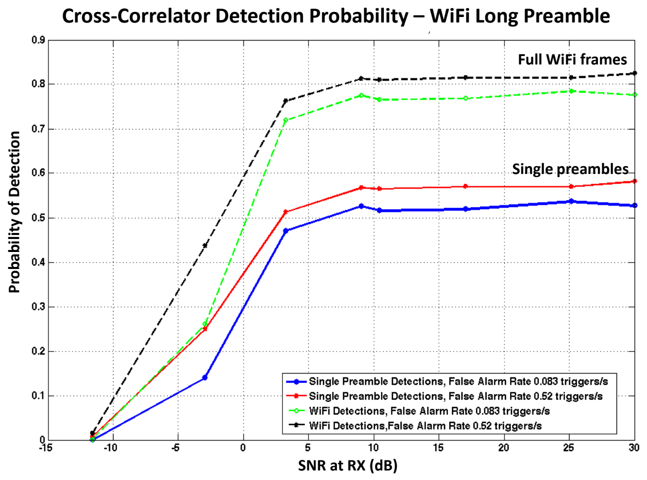


Figure 6: Cross-correlation based detection of WiFi long preamble for packets with a single long preamble as well as full WiFi frames.

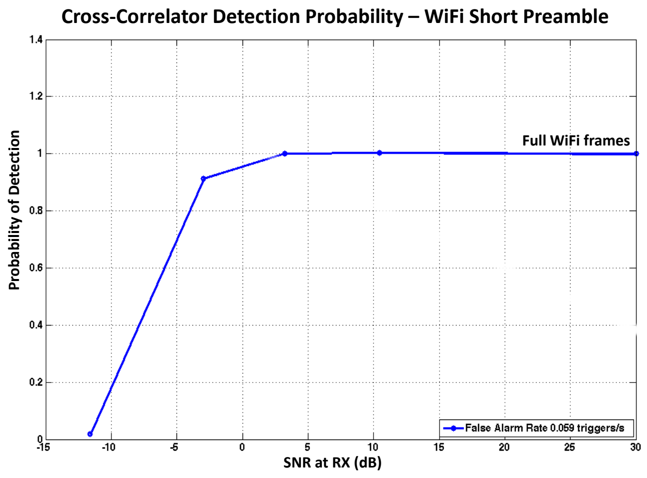


Figure 7: Cross-correlation based detection of full WiFi frames using short preambles.

detection occurs. When the SNR is between -3 dB and 8 dB, the signal level approaches the noise floor, and multiple energy high detections are recorded per frame. The excessive detections are caused by continuous dynamic range variations during the duration of the PHY frame, as a result of the superposition of OFDM signals and noise at roughly the same power levels. This effect fades as the SNR raises above the energy detection threshold, and the energy differentiator reliably produces a single detection per frame for SNRs above 10 dB.

These characterization results demonstrate the functional and efficient performance of the detection system within our framework. Multiple detection methods can be combined to create even more reliable detection mechanisms.

4. VALIDATION ON WIFI NETWORKS

We perform system tests to validate our platform's reactive jamming capabilities with real-time traffic in two different standards: WiFi 802.11g and mobile WiMAX 802.16e. This section presents our experimental results in WiFi 802.11g

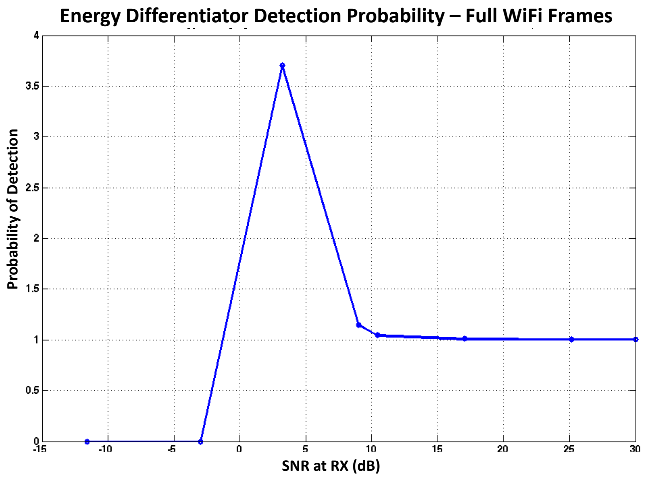


Figure 8: Performance of the energy differentiator in terms of detection probability for full WiFi frames.

networks, while section 5 shows our results in mobile WiMAX networks.

4.1 Experimental Setup

In order to test the effectiveness of our real-time reactive jammer in a non-disruptive controlled manner, we set up a wired, 5-port interconnect network using power splitters as shown in Fig. 9. 20dB attenuators are placed on ports 1 and 2 to emulate the path loss caused by wireless environments and to prevent receiver saturation. A variable attenuator is added to Port 4 in order to provide a large dynamic range for the effects of the jammer on the network. The network is characterized using a vector network analyzer at the numbered ports, and port to port losses are recorded in Table 1.

For this experimental setup, we connect system components as shown in Fig. 9. A Linksys WRT54GL router, programmed with custom firmware, acts as a wireless network access point (AP) and is connected to port 1. The wireless client is placed at port 2. Ports 4 and 5 are dedicated to the jammer transmitter and receiver, respectively. All active system components are programmed to use the same WiFi channel 14 at 2.484 GHz and WiFi protocol 802.11g. No other interference is observed in the channel. Finally, port 3 is connected to an oscilloscope in order for us to observe the signaling environment in the time domain.

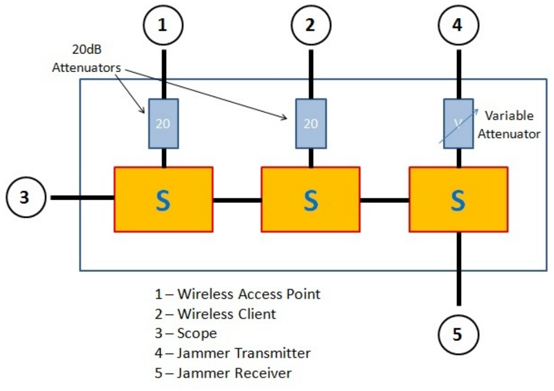


Figure 9: 5-port network showing the connections to the system.

4.2 Experimental Procedure

The AP is connected to a computer running connectivity tests along with throughput traffic tests using *iperf*, a popular network bandwidth measurement tool at application layer. An identical computer is also set up on the wireless client side in order to generate and characterize the wireless traffic between the two users. We run simple ping tests to ensure active network connection and delay. The main experiment is a detailed iperf UDP bandwidth test. We designate the AP as the iperf server, and the wireless client as the iperf client. UDP bandwidth tests with maximum bandwidth of 54 Mbps are conducted repeatedly for 60 second intervals between these two hosts under different jammer settings. The 802.11g buffering parameters and rate back-offs are not constrained and therefore considered as inherent parts of the hardware limitations and 802.11 link characteristics, respectively.

A USRP N210 FPGA in full-duplex operation is introduced into the system as the jamming entity. The jammer is connected to the 5-port network as described in Fig. 9. Its settings is controlled on the fly using the interface presented in Section 2.5. The performance of our reactive jammer is measured and compared against the no-jamming case results along with the continuous jamming case results.

4.3 WiFi Reactive Jamming Results

Using the same hardware platform, we characterize three different types of jamming: a continuous jamming, reactive jamming with relatively long uptime after trigger (0.1 ms), and reactive jamming with short uptime after trigger (0.01 ms).

Bandwidth Under Jamming: The results from our iperf 60-second UDP bandwidth test is shown in Fig. 10. We plot the achievable UDP bandwidth reported by iperf against the signal-to-interference power ratio (SIR) at the AP. A wide SIR range is generated by controlling jammer TX power as well as though the use of stacked attenuators. Note that the SIR axis is arranged in descending order to illustrate the drop in system performance with increase in interference power.

Though we specified the maximum UDP bandwidth at 54 Mbps for the iperf application, the maximum achieved UDP bandwidth during application runs, with or without the jammer, was around 29 Mbps. This is due to overhead within the 802.11 standard, the TCP/IP stack, and additional data overhead of the iperf application itself. The dashed line in Fig. 10 represents maximum achievable band-

	Output				
Input	1	2	3	4	5
1	-	-51.0 dB	-25.2 dB	-38.4 dB	-39.3 dB
2	-51.0 dB	-	-31.7 dB	-32.0 dB	-32.8 dB
3	-25.2 dB	-31.7 dB	-	-19.1 dB	-19.9 dB
4	-38.4 dB	-32.0 dB	-19.1 dB	-	-
5	-39.2 dB	-32.8 dB	-19.8 dB	-	-

Table 1: Insertion loss values measured at the ports of our 5-port network.

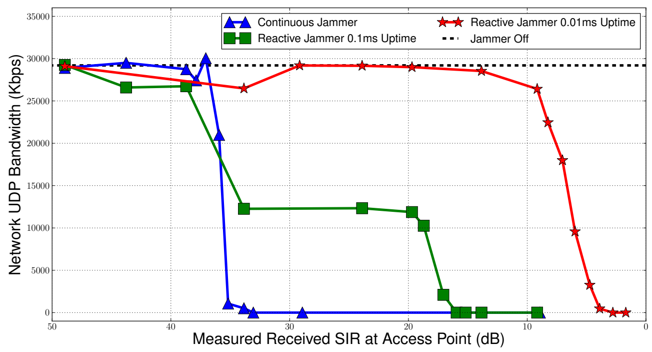


Figure 10: WiFi UDP Bandwidth reported by iperf. The jamming power increases from left to right.

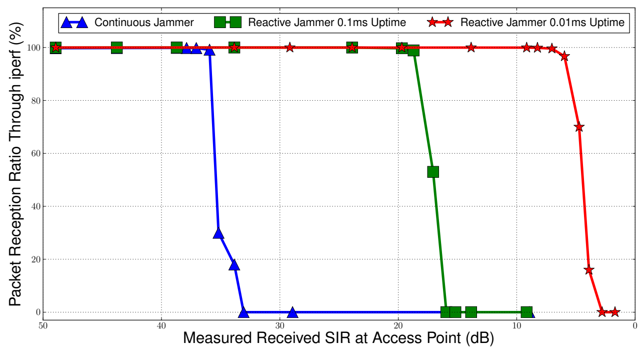


Figure 11: WiFi Packet Reception Ratio reported by iperf. The jamming power increases from left to right.

width without the jammer. A continuous jammer, even at low jamming power, raised the system noise level and significantly reduced network bandwidth. At an SIR of 33.85 dB, the continuous jammer caused the wireless bandwidth to drop to 0 Kbps, and connection to the access point was lost.

Reactive jammers disrupt the wireless networks in a more subtle fashion, and thus are harder to detect. The SIR values presented here depict the channel conditions experienced by the AP during those brief moments when the jammer was actively transmitting. Throughout these experimental runs, the access point had no knowledge of the jammer's presence and always reported an "excellent" link condition. We see from Fig. 10 that a reactive jammer with longer uptime after trigger tends to be more disruptive to the wireless network. The *0.1ms uptime* reactive jammer reduced the network bandwidth by half at an SIR of 33.85 dB, and completely shut down the network at SIR 15.94 dB. The *0.01ms uptime* reactive jammer needed to transmit at a much higher power, yielding an SIR of 2.79 dB at AP, in order to shut down the wireless channel.

Packet Reception Ratio Under Jamming: Figure 11 shows the packet reception ratios (PRR), i.e. the reliability of the link, under various jamming conditions. We experienced link drop-out with a continuous jammer when the PRR dropped from 100% to 0% at around 33 dB SIR (i.e., very low jamming power). Note that though the instantaneous power requirement is low, the continuous jammer must remain on the entire time to shut down the wireless network. The *0.1ms uptime* reactive jammer required 17 dB more instantaneous power to achieve 0% PRR at 16 dB SIR and below. However in this case, the jamming burst only lasted for 0.1*ms*. The *0.01 ms uptime* reactive jammer was the most discreet but also required the highest instantaneous power, yielding 0% PRR only at SIRs below 3 dB.

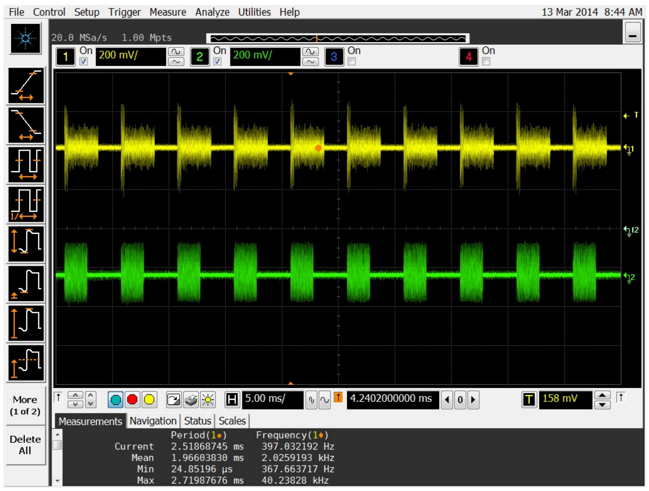
While the results indicates that higher instantaneous jamming powers are required to perform reactive jamming operations, it is important to note that the actual energy requirements are considerably lower. Only a short reactive jamming burst is required to disable the wireless link and force a reset of the client connection to re-establish communications.

Platform Reconfigurability: Our WiFi jamming experiments also demonstrate to a large extent the runtime reconfigurability of our jamming platform. All three different types of jammers were realized at runtime on a single hardware instantiation. We did not have to reprogram the FPGA to switch between different types of jammers. On-the-fly jamming personalities can be changed with a small latency equivalent to the latency of the UHD user setting bus (hundreds of *ns*).

5. VALIDATION ON WIMAX NETWORKS

To demonstrate our reactive jamming platform's ability to reactively jam signals from mulitple high-speed wireless standards, we attempt to detect and jam downlink signals from a mobile WiMAX base station. The target communication standard is 802.16e, using OFDMA as the physical layer. We use an Airspan Air4G macro cell base station to continuously broadcast the downlink signals. The base station is set to operate in Time Division Duplexing (TDD) mode and utilize a 10 MHz bandwidth channel at 2.608 GHz center frequency. In this mode, the hardware sampling rate is set to 11.4 MHz, and the modulation FFT size is set to 1024. Three different preamble carrier sets are defined with different allocations of subcarriers. Each preamble set has a non-zero pilot tones every 3 subcarriers, and 86 guard band subcarriers on each side of the spectrum. Furthermore, each preamble set uses a different 284-value PN sequence to modulate its subcarriers. The preamble carrier set is selected based the *Cell ID* and *Segment ID* value of the base station. For this experiment, we set the base station to Cell ID 1 and Segment ID 0. In the time domain, the WiMAX preamble constitute a single OFDMA symbol at the beginning of each frame, lasting for 100.8μs. Internally, this preamble contains an orthogonal code of 284 samples that repeats itself 3 times within the preamble time. The total duration of this code is 25μs.

Lacking a functional WiMAX receiver to establish a full TCP/IP stack for running system throughput tests with the base station, we only evaluate reactive jamming performance at the physical layer by observing WiMAX and jamming signals on an oscilloscope. The results presented here simply show an informative proof, rather than a thorough measurement analysis. Figure 12 shows a scope capture of both the WiMAX base station signal and our reactive jamming signal in the time domain. In the correlation scheme, we attempt to detect the WiMAX preamble using our 64-sample cross-correlator, running at 25 MSPS. Again, the 25μs orthogonal code in the preamble is being correlated across its first 2.56μs. Insufficient correlation time leads to a misdetection rate of about 2/3 of the packets. However, when



```
             Period(1•)        Frequency(1•)
   Current   2.51868745 ms     397.032192 Hz
      Mean   1.96603830 ms     2.0259193 kHz
       Min   24.85196 µs       367.663717 Hz
       Max   2.71987676 ms     40.23828 kHz
```

Figure 12: Reactive jamming of WiMAX downlink packets from an Airspan Air4G base station.

combining the cross-correlator with the energy differentiator for detection, our system is able to detect reliably 100% of all downlink packets from only the mobile WiMAX network being observed. The lower portion of Fig. 12 shows our jamming signal in real time with a one-to-one correspondence to the WiMAX downlink frames. With the energy differentiator alone, we can simply detect wireless activities without the ability to pinpoint the underlined wireless technology. Having both effective detection and protocol awareness can enable a wide range of sophisticated attacks, such as a type of "surgical jamming" mentioned above, as well as malicious wireless packet injection to interfere with ongoing communications.

Limitations of the platform: it is worth mentioning that our reactive jamming platform has some inherent limitations, as clearly demonstrated in this section. The signal detection system works on a fixed correlation window (64 samples) and digital sampling rate (25 MSPS), thus making it difficult to detect signal preambles of long duration or different sampling rates.

6. CONCLUSIONS

In this paper we have presented our initial implementation of a real-time, protocol-aware, reactive jammer targeted for high-speed wireless networks with preambles. Based on the popular SDR platform USRP N210, our jammer is highly versatile and adapts quickly to intercept 802.11 and 802.16e network traffic under various channel conditions.

Our initial experimental results show that effective reactive jamming in high-speed wireless networks is indeed feasible for an adversary. The results also show that our platform can reliably detect and react to packets from a range of wireless standards. The platform is extremely flexbile and programmable to adapt quickly on the fly. This tool sets an example to prove that SDR-based reactive jamming is practical and should be considered a serious physical layer security threat that warrants additional research and consideration in standard development. The testbed presented in this paper can be an effective tool for studying and developing countermeasures to a new series of real-time over-the-air physical layer attacks.

7. ACKNOWLEDGMENTS

The research presented in this paper was based upon work supported by the National Science Foundation under Grants CNS 1228847, ECS 1028608, CNS 0854946, and CNS 0923003.

8. REFERENCES

[1] GNU Radio. `http://gnuradio.org/redmine/projects/gnuradio`.
[2] WARP Project. `http://warpproject.org`.
[3] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa. On the Performance of IEEE 802.11 under Jamming. In *Proc. of INFOCOM*, volume 0448330, pages 1265–1273, Apr. 2008.
[4] J. Chen, S. Sen, M. Chiang, and D. J. Dorsey. A Framework for Energy-efficient Adaptive Jamming of Adversarial Communications. In *Proc. of CISS*, 2013.
[5] S. Gollakota and D. Katabi. iJam : Jamming Oneself for Secure Wireless Communication. Technical report, MIT, 2010.
[6] S. Gollakota and D. Katabi. Physical layer wireless security made fast and channel independent. In *Proc. of IEEE INFOCOM*, pages 1125–1133, Apr. 2011.
[7] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. In *Proc. of IEEE INFOCOM*, pages 1307–1315, 2007.
[8] Y. Liu and P. Ning. BitTrickle: Defending against broadband and high-power reactive jamming attacks. In *Proc. of IEEE INFOCOM*, pages 909–917, Mar. 2012.
[9] T. Pongthawornkamol and K. Nahrstedt. Alibi Framework for Identifying Reactive Jamming Nodes in Wireless LAN. In *Proc. of IEEE GLOBECOM*, Dec. 2011.
[10] S. Prasad and D. J. Thuente. Jamming attacks in 802.11g - A cognitive radio based approach. In *Proc. of IEEE MILCOM*, pages 1219–1224, Nov. 2011.
[11] W. Shen, P. Ning, X. He, and H. Dai. Ally Friendly Jamming: How to Jam Your Enemy and Maintain Your Own Wireless Connectivity at the Same Time. In *Proc. of IEEE Symposium on Security and Privacy*, pages 174–188, May 2013.
[12] I. Shin, Y. Shen, Y. Xuan, M. T. Thai, and T. Znati. A Novel Approach Against Reactive Jamming Attacks. *Journal of Ad Hoc & Sensor Wireless Networks*, 12(1-2):125–149, 2011.
[13] D. J. Thuente and M. Acharya. Intelligent jamming in wireless networks with applications to 802.11b and other networks. In *Proc. of IEEE MILCOM*, pages 1075–1081, 2006.
[14] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Reactive Jamming in Wireless Networks - How Realistic is the Threat? In *Proc. of ACM WiSec*, pages 47–52, 2011.
[15] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proc. of ACM MobiHoc*, pages 46–57, 2005.