

A System Level Solution for DSA Systems: from Low-Cost Sensing to Spectrum Database

Osama A. H. Al Tameemi and Mainak Chatterjee
Electrical Engineering and Computer Science
University of Central Florida
osama, mainak{@eecs.ucf.edu }

Abstract—In this paper, we present and implement a low-cost yet effective architecture that enables dynamic spectrum access (DSA) for any type of network. Our intention is to break the cost-scalability barrier and show that a complete system level solution for a database-assisted DSA system can be implemented with standard servers and inexpensive software configurable RF chips, thereby achieving economics of scale.

First, we present the overall architecture that is capable of providing networks of any size to perform in-band and out of band channel access in a dynamic manner. The two main components of this architecture are the received signal strength indicator (RSSI) sensing network and the DSA server. For the RSSI sensing network, we built wired and wireless spectrum sensors that operate on 280–930 MHz using low-cost off the shelf software configurable RF (SCRf) chips. To get the RSSI values on a set of bands, we use generic micro-controllers to program the operating parameters (scan range, center frequency, bandwidth resolution, demodulation scheme and scan rate) of the SCRf chips. The wireless sensors transmit the sensed RSSI values to the nearest Ethernet-enabled hub using a light-weight communication protocol. The hub aggregates the data from multiple sensors and streams to the DSA server using UDP over IP. On receiving the real-time RSSI values from various sensors, the DSA server stores them in database engine with other meta data. Entries from the database are used by the channel allocation service that finds the best channel for the inquiring DSA nodes. To demonstrate the efficiency of the implemented database-assisted DSA system, we compare it to a legacy system and show the benefits in terms of resilience to jamming, channel relinquishment on primary arrival, and best channel determination and allocation. We also show the performance gains in terms of frame error rate (FER) and spectral efficiency. Finally, we compare the RSSI sensitivity of the low-cost sensors to that of a professional spectrum analyzer.

Index Terms—Dynamic spectrum access, sensing, spectrum database, testbeds.

The recent years witnessed an increasing demand for wireless spectrum. Such demand is being answered by initiatives that are embracing a more flexible bandwidth allocation and usage model. For example, the FCC’s technological advisory council named spatial frequency reuse as the “best practice” for alleviating the bandwidth scarcity problem [4]. Accordingly, there has been a paradigm shift from static spectrum allocations towards more ‘liberalized’ notion of dynamic spectrum access (DSA) where secondary users (i.e., unlicensed users) can “borrow” idle spectrum from primary users (i.e., spectrum licensees), without causing harmful interference to the latter.

To realize DSA, reliable sensing has to performed (whether

by dedicated scanners or by the DSA nodes themselves). Unfortunately, sensing is often the most expensive part of the DSA system and has a significant economic impact on the deployment of such systems [3] as shown the International Telecommunication Union (ITU).

In order to provide a consistent assay on spectrum vacancies, and also to have less reliance on sensors for sensing, a database driven spectrum allocation and access policies has been mandated [1]. However, even with such approach, having *enough* spectrum sensors to feed the database (and the derived services like the radio environment map) remains a bottleneck from a scalability perspective as such approaches rely on expensive hardware platforms for the sensing tasks [2].

Although many techniques exist for spectrum sensing like matched filter, energy detection (ED), cyclostationary, etc [5], [6], [12]; energy detection stands out as one of the optimum methods as it is simple and does not require any prior knowledge about the primary user(s) [11]. Simply stated, ED revolves around measuring the amount of energy on a specific bandwidth. Typically, heavy duty hardware devices (SDRs, spectrum analyzers) are employed for this simple task which is an overkill. Also, the cost of these devices becomes a bottleneck for large scale deployment. We argue that these light-weight sensing tasks can potentially be done on relatively inexpensive and computationally light devices.

Our work is motivated by the proliferation of low cost off-the-shelf software configurable RF (SCRf) chips. Popular examples are the CC2500, CC1100, CC2550 by Texas Instruments [14] and RFM22, RFM23, and RFM69 by Hop-erf [15]. These devices have built-in capability to record received signal strength indicator (RSSI) values (digital or analog)— a feature which we believe has been overlooked for its potential for spectrum sensing, particularly when combined with the energy detection algorithms like localization algorithm double-thresholding (LAD), forward consecutive mean excision (FCME), etc [8], [13]. Our design philosophy is based on i) *harnessing the inexpensive commercial off-the-shelf SCRf chips* that offer this RSSI feature, instead of traditionally used SDRs and spectrum analyzers allowing for a low-cost solution to sensing, scalability, and cost, and ii) *realizing flexible system reconfigurability* via user-friendly programming interface. Though the spectrum analyzers and SDRs are far more capable than the SCRf chips for spectral analysis, however for ED, such features are irrelevant. Most

importantly, the SCRF chips sold for \$3-\$4 are orders of magnitude economical than SDRs and spectrum analyzers.

In this paper, we present and implement a DSA system architecture that primarily has two main components. i) the spectrum sensing network and ii) the DSA server— each of which has several sub-components. As for spectrum sensing, we use software configurable RF (SCRF) chips, not to be confused with Software Defined Radios (SDRs) [16]. SCRF chips are equipped with a list of operational configurations which we program using software to measure the RSSI on desired frequencies. We implement the DSA server in the form of Java-based modules on a Linux box to: i) collect RSSI readings from the sensing network, ii) process and store RSSI readings in the database, and iii) use the database entries to allocate the best channels to the inquiring DSA nodes. We compare the performance gains of the proposed DSA system against legacy wireless systems in terms of best channel selection, resilience to jamming, frame error rate (FER) and spectral efficiency.

I. SYSTEM ARCHITECTURE

The proposed DSA system architecture, which has also been implemented, is shown in Fig. 1. The overall system architecture consists of the RSSI sensing network and the DSA server. The sensing network captures the RSSI readings and streams them real-time to the DSA server where the data is received, parsed, stored in a database, and used to select the best channel for the inquiring DSA nodes.

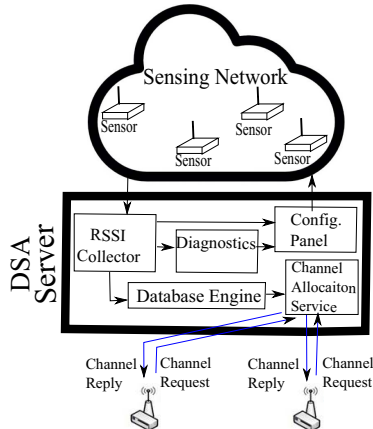


Fig. 1. System Architecture.

The sensing network consists of low-cost spectrum sensors which are capable of streaming real-time RSSI measurements (wired or wirelessly) to the DSA server. The RSSI collector in the DSA server continuously receives the streams of data from multiple sensors and passes them to the database engine which creates a record (that includes time-stamps, sensor ID, sensor location) for each reading. The configuration panel module allows changes to the radio and network parameters for the sensing network based on the real-time performance diagnostics. Entries from the database are used by the the channel

allocation service to provide the end-user (DSA nodes) with the best channel in their vicinity.

II. RSSI SENSING NETWORK

The sensing network consists of spectrum sensors which can be wired or wireless. We must first understand the building blocks of the sensor nodes.

A. Spectrum Sensors

A spectrum sensor unit/node primarily consists of two main components: the RF-front end and the controller. Depending on how the sensor would be connected to the rest of the network, there could an additional Ethernet interface. A block diagram along with what has been actually implemented is shown in Fig. 2. We emphasize that the design is not based on any specific component. These components are treated as generic building blocks. Our goal is to provide a guideline on how to capitalize on the off-the-shelf SCRF chips and controllers for making low-cost spectrum sensors.

1) *RF front-end* : The RF front-end is responsible for measuring the RSSI values on the target bands. The RF front-end makes use of the RSSI reporting feature that is available in the off-the-shelf SCRF chips. As discussed earlier, these SCRF chips can be configured as transmitters (Tx) or receivers (Rx) in addition to configurable modulation schemes, bandwidth, and data rates. The features of any specific SCRF chip dictate the RF properties and capabilities of the sensor. For example if the SCRF chip uses an analog technology with maximum bandwidth of 500 KHz, then the sensor will have an analog front end with 500 KHz as the maximum bandwidth. If needed, the RF front-end can also be used to transmit the measured RSSI values over a radio channel as will be shown later in Section II-C.

2) *Controller*: In order to configure and control the SCRF chips, a controller is needed— the interface of which depends on the SCRF chip in question. The serial peripheral interface (SPI) is a popular choice for TI and HopeRF chips. The controller instructs the SCRF chip with the RF configuration parameters (target frequency, bandwidth, scanning rate and resolution) based on the user inputs. The RSSI values from the RF front-end are then read by the controller. Depending on the network architecture, the controller either i) instructs the Ethernet interface to stream the RSSI values to the DSA server using some real-time streaming protocol, or ii) reconfigures the RF-front end as a transmitter to transmit the RSSI value over a radio channel.

3) *Ethernet Interface*: This optional interface can be used to make use of an existing network infrastructure to stream the RSSI values, thus relieving the RF-front end for sensing only. It can be noted that, the Ethernet circuitry could be integrated with the controller or it could be external to the controller.

B. Network Architecture

Though the Ethernet-enabled sensors can stream the RSSI values directly to the DSA server, the wireless sensors cannot do so as the DSA server is not wireless enabled. A wireless enabled DSA server does not help because of i) additional cost

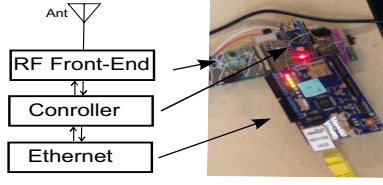


Fig. 2. Block diagram of sensor node (left) with the corresponding implementation (right).

due to specialized hardware, ii) reduced sensing area as the sensors will have to be able to directly communicate with the DSA server, and iii) lack of support for multiple frequencies.

In order for the *wireless* sensors to send their data to the DSA server, the sensors must send the data to a wireless hub which will then send the data to the DSA server via Ethernet as shown in Fig. 3. The figure also shows the Ethernet-enabled sensors that stream directly to the DSA server over the Internet. Apart from the hub and the sensors streaming data over the Internet, they could also be directly connected to the DSA server. Both configurations are illustrated in Fig. 3.

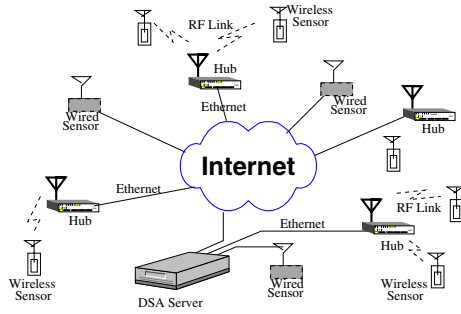


Fig. 3. Overall network architecture showing how wireless sensors, hubs, and Ethernet-enabled sensors connect to the DSA server.

C. Network Operations

Let us now elaborate on how the various network components communicate over RF and Ethernet.

1) *Wired Sensors*: Recall, a wired sensor has an RF-front end, a controller, and an Ethernet interface.

On power-on, the sensor contacts the DSA-server over Ethernet to obtain the list of frequencies that need to be scanned including bandwidth, sweep time, and scan resolution (i.e., scan-list). Once the scan-list is obtained, the controller cycles through the list one frequency at a time and instructs the RF front-end to measure the RSSI on each frequency. The controller then fetches the RSSI value from the RF front-end and passes it to the Ethernet interface where the controller instructs the Ethernet interface to build a UDP packet to be streamed to the DSA server. The payload of the UDP packets can carry: i) single RSSI value or ii) multiple RSSI values– the decision of which depends on the time critically of RSSI reporting. Apart from the RSSI value(s), the payload contains other information such as nodeID, frequency, sensor coordinates.

2) *Wireless Sensors*: A wireless sensor uses the *same* RF front-end for sensing and for transmitting the sensed data to the hub. On a broadcast channel, the hub instructs each wireless sensor to scan a specific set of frequencies. It also manages the channels (i.e., uplink frequency and the transmit time slot) for each sensor. This switching between sensing and transmitting by the RF front-end is achieved by the controller. It starts with the controller instructing the RF front-end to listen to the dedicated broadcast channel (i.e., Rx mode). Once the broadcast is received, the controller extracts the scan-list and instructs the RF-front end to measure the RSSI values. Similar to the wired sensors, the controller fetches the RSSI values and switches the RF front-end in the Tx mode and transmits the data to the hub. Once done, it switches back to the Rx mode and the process repeats.

3) *Hubs*: The role of a hub is not only to communicate with the wireless sensors in a “hub and spoke” configuration but also to serve as a wired sensor. The hub is equipped with the RF-front end, controller and Ethernet interface. While sensing, it simply behaves just like the wired sensors. While controlling the wireless sensors, the hub is responsible for i) broadcast of scan list to respective sensors (i.e., task allocation) ii) uplink channel allocation to individual wireless sensors, and iii) data aggregation and UDP streaming.

On power-on, the controller contacts the DSA server and obtains a scan list for itself and the wireless sensors connected to it. As in the wired sensors, the controller cycles through the list of frequencies, configures the RF front-end, fetches the data and streams it over UDP to the DSA server. Next, it configures its RF front-end for transmission on the broadcast channel and transmits the scan-list to the wireless sensors. It can be noted that the hub might instruct *all* the wireless sensors to scan the same set of frequencies or it might instruct the sensors to scan different sets of frequencies.

Once done, the hub switches the RF front-end to the receive mode and waits for the wireless sensors to send their data. Transmissions by the wireless sensors are scheduled by the hub using any mechanism such as polling or token passing. Depending on the time-criticality of the RSSI measurements the hub can i) have a single RSSI measurement per packet, ii) aggregate multiple RSSI measurements from the same wireless sensor into a UDP packet, or iii) aggregate multiple RSSI measurements from different wireless sensors into a UDP packet. In all three cases, it streams the UDP packet to the DSA server using its Ethernet interface.

III. DSA SERVER

As discussed in the system architecture, the DSA server has five primary components: RSSI collector, real-time diagnostics, configuration panel, database engine, and the channel allocation service.

RSSI Collector: The RSSI collector can be a single or multi-threaded server depending on the number and size of RSSI incoming UDP streams from the sensors and hubs. The RSSI collector extracts all the fields (i.e., nodeID, location, freq,

bandwidth, and RSSI values) from the UDP packets and timestamps and passes them to the database engine. The same information is also passed to the real-time diagnostics module. Apart from the RSSI streams, the RSSI collector also gets the queries for the scan-list from the sensors and hubs which is directed to the configuration panel.

Real-time Diagnostics: Diagnostics allow spectrum managers to have real-time assessment of the spectrum assay of the RF field being sensed. Various representation techniques and/or display visualization can be used that best conveys the meaning of the parameters captured by the sensing network. Based on the assessment, it also provides a feedback to the configuration panel for scan list parameters update.

Configuration Panel: The configuration panel is used to configure the scan parameters for the sensors and hubs. Such configuration can be done in an automated manner in response to what the diagnostics are or it could be done manually by the end-user. In either case, these parameters (target frequencies, bandwidth, sweep time, and scan resolution) are sent to the sensors and hubs.

Database Engine: This is a multi-threaded server which receives the time-stamped entries from the RSSI collector. This engine provides a mechanism for storage and retrieval of data by storing data in a NoSQL format, primarily because of its simplicity and scalability. If needed, the engine purges entries which are older than a specified time period. It also replies to all the queries from the channel allocation service.

Channel Allocation Service: For any channel allocation and access technology, it is important for a pair of transceivers to know the channel that provides the best possible data rate. This is even more crucial in a DSA system because of the fleeting nature of the set of available channels. The channel allocation service is a routine that makes use of the database entries to find the best available channel for the given locations of the transmitter and the receiver. As the database might not contain the RSSI values for requested locations, channel allocation service resorts to some interpolation/extrapolation methods to estimate the data rate for the given locations.

IV. HARDWARE AND SOFTWARE IMPLEMENTATION

Capitalizing on the SCRF chips' capabilities, we implemented a low-cost system level solution for dynamic spectrum sharing. For the sensing network, we designed and built sensors that worked as i) wired sensor, ii) wireless sensor, and iii) hubs. The DSA server was implemented on a Linux machine.

A. Wired Sensors

Our wired sensor is meant to detect sub-GHz frequencies. All the components were low-cost off-the-shelf devices—the total cost of which was less than \$10. The individual components are shown in Fig. 5(a) while the assembled one is shown in Fig. 5(b). The sensor encapsulates the RSSI values in UDP packets and stream them to the DSA server using the

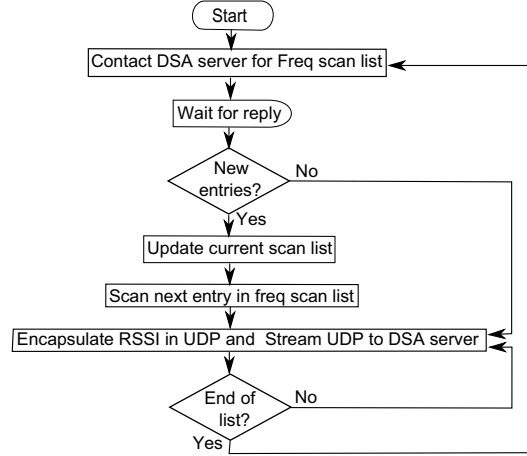


Fig. 4. Operation procedure for the wired sensor.

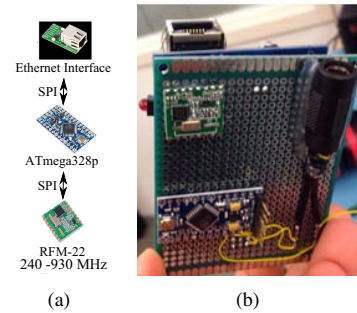


Fig. 5. (a) Components for wired sensor; (b) Assembled wired sensor.

following frame format:

nodeId:targetFreq:BW:RSSI:XX:YY

The *nodeId* is used to identify the sensor, the *targetFreq* and *BW* are the center frequency and bandwidth being scanned, *RSSI* is the measured signal strength in dBm, *XX* and *YY* are the X-coordinate and Y-coordinate of the sensor. The operational procedure for the wired sensor is shown in Fig. 4.

RF Front-end: The RF front-end for this sensor is RFM22 by HopeRF [15] which has a frequency response from 240 MHz to 930 MHz. The receiver sensitivity is -118 dBm with a configurable receiver bandwidth from 2.6 KHz to 620 KHz. It also has an RSSI reporting feature. The *transmit* capabilities of RFM22 are exploited for the hub and the wireless sensors as will be discussed in sections IV-B and IV-C.

Controller: We used the microcontroller ATmega328p as the controller. It is an 8 bit device with 32 KBytes of ROM, 2 KBytes of RAM, with 20 MIPS. Packet encapsulation and streaming were done as follows.

Ethernet Interface: We used W5100 chip which is a hardwired TCP/IP embedded Ethernet controller that enables easier Internet connection for embedded systems. An attractive feature of this chip is that it can be controlled via SPI by the ATmega328p microcontroller.

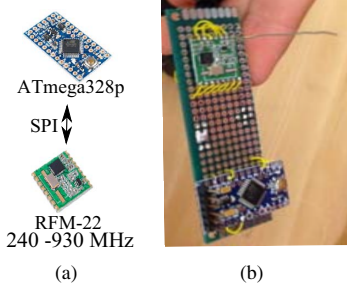


Fig. 6. (a) Components for wireless sensor; (b) Assembled wireless sensor.

B. Wireless Sensors

The wireless sensors have the same RF front-end and the controller as that of the wired sensor. Being wireless, it does not have Ethernet connectivity. For transmission, it uses the same RF front-end that is used for sensing. For the Tx part, the RFM22 supports FSK, GFSK, and OOK modulation schemes with data rates from 1 to 128 Kbps. It has configurable transmit power from 8 dBm up to 17 dBm. The wireless sensors send data to the hub using a customized light-weight protocol using the following format:

`MyNodeID : targetFreq : BW : RSSI : XX : YY`

The `MyNodeID` is the id of the sensor that measured the RSSI, `XX` and `YY` are the x and y coordinates of the sensor, `targetFreq` is center frequency of the band being sensed, `BW` is the bandwidth, and `RSSI` is the measured value. The components of a wireless node are shown in Fig. 6(a) while an assembled wireless sensor is shown in Fig. 6(b).

C. Hub

We used the same hardware components that we used for the wireless sensor to serve as the hub for the wireless nodes. In addition to the functions that are performed by the wired sensor, the hub has been programmed to query the wireless nodes, manages their channel access using a polling-based MAC, get their responses, encapsulate the data in UDP packets, and stream them to the DSA server.

Using a dedicated channel, the hub queries the wireless nodes in its vicinity using the frame format:

`YourNodeID : targetFreq : BW`

where The `YourNodeID` is the id of the sensor that is being queried by the hub, `targetFreq` is center frequency of the band to be sensed, and `BW` is the bandwidth. Once the hub receives the data from a wireless sensor, it encapsulates the entire data in a UDP packet and streams it to the DSA server.

D. DSA Server

The DSA server, consisting of the five modules, is the core of the entire system. We implemented all the modules in Java on a Linux box running Ubuntu 14.04.

For the RSSI collector, a multi-threaded server is run on port 25100 (randomly chosen) to read the incoming UDP packets. To infer the presence of primaries, we implemented

the classical periodogram in Java that essentially compares the RSSI values against a threshold. The diagnostic functionality keeps track of the active primaries and their spectrum usage patterns. A multi-threaded UDP server was implemented to allow the sensors in the field to open a UDP connection to the configuration panel. This connection was used by the sensors to request the scan-list from the configuration panel and also for the configuration panel to respond with the same. The UDP frame format that was used was:

`nodeID : freq1 : BW1 : freq2 : BW2 : ... : freqN : BWN`

with the `nodeID` being the id of the node asking for the scan-list, BW_i is the bandwidth for the i -th channel around the target center frequency $freq_i$. Using the configuration panel, wired sensors were configured to scan channels with bandwidth of 300 KHz centered at 900, 905 and 910 MHz. The RSSI reported value, `nodeID`, node's location, bandwidth, target frequency from all sensors were time-stamped and stored as 'Comma Separated Values' (CSV) file format in the database engine as:

`a, b, c, ...`

We implemented the *channel allocation service* as a multi-threaded Java server running on port 25200. This service is used by the DSA enabled transmitter-receiver pairs as they seek the best vacant channel in their vicinity. Upon receiving a channel allocation request with the format

`nodeId : XX : YY`

the channel allocation service extracts the pairs' location (i.e., (x, y) coordinates). It then accesses the database engine to find entries that might have been recorded by sensors in the vicinity of (x, y) . We used a modified version of Shepard's interpolation technique to estimate the RSSI (and hence the best possible channel) for the location in question by fusing the RSSI values recorded by sensors in the vicinity [9]. In particular, we implemented the technique proposed in [7] which was computationally lightweight and also efficient for different types of primary networks. The service responds back with a UDP packet using the following format:

`nodeId : freq : BW`

where `freq`, `BW` are the frequency and bandwidth of the recommended channel.

V. PERFORMANCE EVALUATION

Let us demonstrate the gains achieved from the implemented DSA system as compared to a legacy system. To do so, we deployed wired spectrum sensors in an indoor environment (laboratory) and programmed to sense 300 KHz of bandwidth centered at 900 MHz, 905 MHz, and 910 MHz. The sensed RSSI were streamed to the DSA server.

To induce controlled interference and mimic the primary users, we built two software configurable jammers using RFM22 transceivers as the front-end and ATmega328p as the controller. We were able to control the frequency, bandwidth, transmission power and transmit pattern of the jammers.

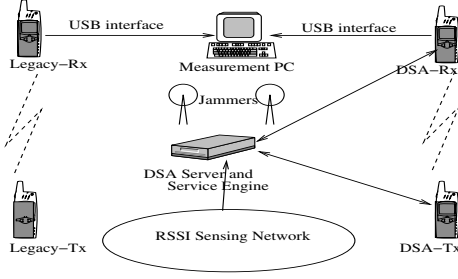


Fig. 7. Experimental setup for DSA and legacy comparison.

A. DSA versus Legacy Tx-Rx

To show the benefits of DSA enabled communications, we built two transmitter-receiver pairs with identical components (i.e., same RF front-end and controller sections). One pair was enabled with DSA capability—the pair was programmed to contact the DSA server (over Ethernet) and utilize the ‘channel allocation service’. The other transmitter-receiver did not have this DSA capability and worked just like legacy radios. For fair comparison, both pairs were placed equi-distant from the jammers. The Tx-Rx distance for both pairs were also the same as shown in Fig. 7.

On start-up, both transmitters were programmed to transmit packets of 16 bits on the default channel of 900 MHz using FSK with transmission power of 2 dBm. Each packet was identified with a sequence number. By default, the receivers of both pairs would listen on the 900 MHz channel. On reception of a packet, the receivers would send an acknowledge (ACK) to the respective transmitter. Both the transmitter and the receiver keep track of the number of packets that are successful.

The DSA pair (both transmitter and receiver) continuously i) contacts the DSA server to check for primary’s presence (similar to the SenseLess framework [10]), and ii) requests for a new channel if there are 5 consecutive packet failures¹. On the other hand, the legacy pair continues to use the 900 MHz channel regardless of packets being successful or not. The receivers of both pairs were interfaced to a Windows machine that used a Java snippet to monitor the receivers’ throughput and packer error rate.

B. Frame Error Rate (FER)

In addition to displaying the throughput and the packet error rate of each receiver, the Java snippet also displays additional information as shown in Figs. 8(a) and 8(b). With time on the x -axis, both figures show which channel is being used. Fig. 8(a) shows that channel 900 MHz is being used followed by 905 MHz, 910 MHz and again 900 MHz. Every time, the channel was changed when the ‘failure’ remained for more than 5 packets (also shown on the figure). On the other hand, the legacy pair continues to use the 900 MHz channel for the

¹We chose 5 arbitrarily; although in real scenarios it will depend on the type of data, mission criticality and other parameters.

entire duration in-spice of the long period of ‘failure’ as shown in Fig. 8(b).

We define a *frame* as a group of 10 packets. Both Figs. 8(a) and 8(b) show small ticks on top of the success field that represent the frame times. We define frame error rate (FER) as the fraction of packets that are unsuccessful in a frame. For example, if 3 packets are unsuccessful in a frame, the FER is 0.3. For the DSA pair, it can be noted from Fig. 8(a) that the FER is 0 for most frames, and either 0.1 or 0.2 for few (due to sporadic noise). However, when channel switching occurs due to the channel being jammed, it can be seen that the FER is considerably high for a small period of time. When the DSA pair switch their communication to a new channel, the FER recovers. For the legacy pair, the FER remains very high for the entire duration the channel is jammed.

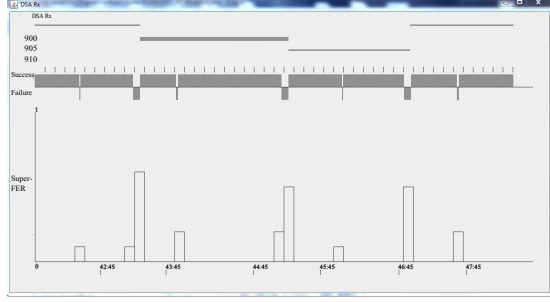
C. Jammer in Operation

Figs. 8(a) and 8(b) show that both pairs start on 900 MHz channel at $t=0$. Till $t=43:00$, packets were received successfully at both receivers with some sporadic failures due to the background noise. Jammer-1 started jamming on 900 MHz at $t=43:02$ as shown in Fig. 9(a). It had a transmit power of 20 dBm. Note how the occurrence of failures at both pairs are correlated with the time at which jammer-1 was activated. For the legacy pair, from Fig. 8(b), it is clear that the pair is completely jammed and it suffers from 100% packet loss (indicated by $FER=1$) for the entire duration jammer-1 was active on the 900 MHz. As for the DSA pair, when jammer-1 caused 5 packet failures, the DSA pair sought a new channel by contacting the DSA server which consulted the RSSI values on all the candidate channels as shown in Fig. 10. Based on the lowest RSSI values, the DSA pair switched to 905 MHz as it was chosen by the DSA server. To further show the DSA system in work, we activate jammer-2 on 905 MHz from $t=45:15$ which forced the DSA pair to contact the DSA server again. As a result, the pair switched to 910 MHz as it was the only available channel at $t=45:018$. Finally, jammer-1 switches the jamming channel from 900 MHz to 910 MHz at $t=46:49$. As a result, the legacy pair is able to resume successful packet reception (on 900 MHz) as indicated by Fig 8(b). In response to jammer-1 jamming on 910 MHz, the DSA pair again contacts the DSA server and switches from 910 MHz to 900 MHz.

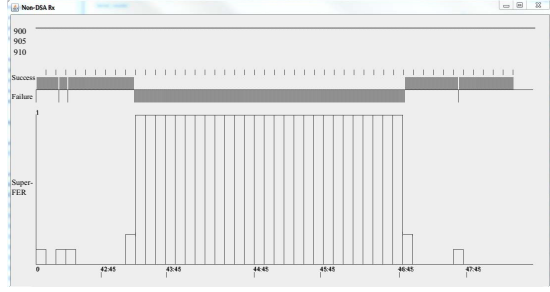
D. Link Sustainability and Spectral Efficiency under Jamming

We compare the link-sustainability and the modified spectral efficiency of both pairs under jamming. Fig. 11(a) shows the percentage of time each pair maintained a valid transmitter-receiver connection. Apparently the DSA pair achieves a higher ratio as it is able to *recover from jamming* by **dynamically switching** to a free channel unlike the legacy pair.

Traditionally, spectral efficiency is defined as the number of bits per second per Hz i.e., it gives the bit-rate between a transmitter-receiver pair per Hz. However, spectral efficiency alone does not reflect what fraction of time was the channel used by the secondary users when the primary was absent.

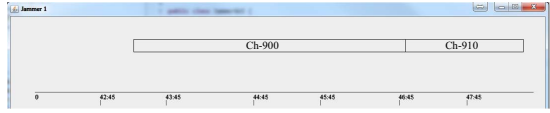


(a)



(b)

Fig. 8. (a) Reception status of the DSA receiver. (b) Reception status of the legacy receiver.



(a)



(b)

Fig. 9. (a) Activity of jammer-1. (b) Activity of jammer-2.

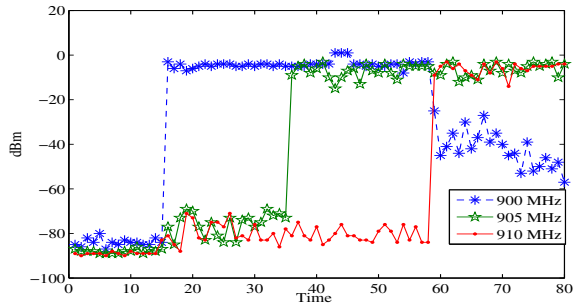


Fig. 10. RSSI values for channels 900, 905, 910 MHz during the evaluation.

To capture the fraction of time a DSA pair utilizes an empty channel, we calculate the *modified bit-rate* by dividing the number of bits transferred by the *vacancy time of the channel*.

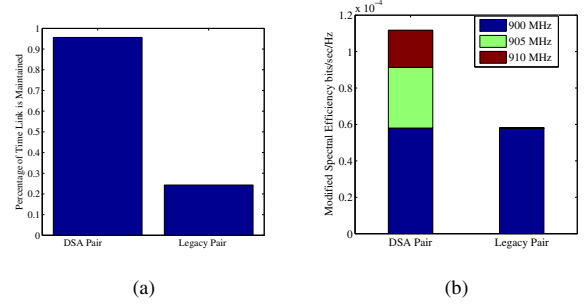


Fig. 11. (a) Link sustainability of DSA and legacy pairs when jammed. (b) Modified spectral efficiency of DSA and legacy pairs with 3 channels with jammers active.

The *modified spectral efficiency* is obtained by dividing the modified bit-rate by the bandwidth of the channel.

The modified spectral efficiency of the DSA and legacy pairs (calculated for the 900, 905 and 910 MHz channels) are shown in Fig. 11(b). It can be noted that the three channels have different modified spectral efficiency values. The 900 MHz channel has the highest while the 910 has the lowest, although 1) the DSA pair has the same instantaneous bit-rate on all of the three channels when they were being used, 2) the channels have the same bandwidth (300 KHz), 3) the DSA pair spent approximately the same amount of transmission time on each of the channels as illustrated in Fig. 8(a). The reason for the difference in the modified spectral efficiency values for the 3 channels is because the channels' vacancy times were different. as seen in Figs. 9(a), and 9(b). For example, the 900 MHz channel was utilized by the DSA pair for the entire duration it was vacant. On the other hand, while the 905 channel was vacant for almost double the time, however the DSA pair used it for about half of the time only (during the other half it was transmitting on the 900 MHz channel).

As for the legacy pair, only the 900 MHz was used. Hence it had 0 bits/sec/Hz from the other two channels (905 and 910 MHz) despite being vacant.

E. Performance Gains

In order to quantify the gains obtained from the DSA enabled transmitter-receiver pair, we compare the normalized throughput of the DSA pair with that of a non-DSA pair as follows. The jammer emulated primary activity with a Poisson distributed 'On-Off' model with average 'On' ratio (also referred to as duty-cycle) ranging from 0.1 to 1.0. The resultant normalized throughput values are shown in Fig. 12(a). For very low primary activity, the legacy pair outperforms the DSA pair as the latter incurs some overhead due to sensing and server access latency. When primary activity increases, the normalized throughput of the legacy pair decreased linearly. However, there was no change in the normalized throughput for the DSA pair as long as there was a channel available.

F. Effects of Primary Average-ON Time on DSA Throughput

Although two identical primary users can have the same duty cycle (say 0.5; i.e., 50% ON and 50% OFF), it can

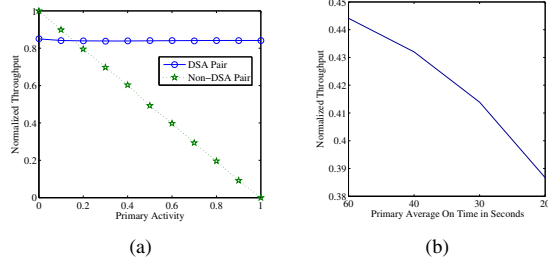


Fig. 12. (a) Normalized throughput of DSA and legacy pairs. (b) Primary average-ON time and resultant DSA throughput.

have different effects on the throughput of the secondary users depending on how frequent the switching between ON and OFF occurs. For the same duty-cycle, a primary with a shorter average ON time, will transmit more often for short periods of time. This affects the secondary users since they incur more overhead i.e., detect the presence of the primary users, refrain from using that channel, continue to contact the DSA server to check when the channel is free again. This overhead reduces the time for which the DSA pair is able to transmit data—thereby reducing throughput. To demonstrate such effect, we had the DSA pair transmitting on the 900 MHz channel for 240 seconds (no primary) and measured the resultant (normalized) secondary throughput. With the duty cycle of the primary user fixed at 0.5, the average ON time was varied. Fig. 12(a) shows the normalized secondary throughput. As expected, shorter average ON times of the primary (frequent ON-OFF switching) lead to more overhead for secondary and hence reduced throughput.

G. Spectrum Sensor Sensitivity

In order to show the practicality of our sensors, we compare the RSSI measurements from our wired sensor against a professional spectrum analyzer by Rohde & Schwarz. We used the R&S FSU model. The sensor and the spectrum analyzer were equipped with the same antennas (for fair comparison) and had the same scanning settings i.e., they scanned the RSSI values on a bandwidth of 300 KHz centred at 900 MHz. Both units were placed next to each other and equi-distant from a jammer. The jammer was set to transmit at power levels of 2, 5, 8, 11, 14, 17, and 20 dBm. The RSSI values from both units were recorded as shown in Table I. In all cases, the wired sensor had higher RSSI values; thus demonstrating better sensing capability. This also means that our low-cost sensors were able to detect the primary signal from a farther distance as compared to the spectrum analyzer; thereby increasing the sensing range and providing more protection for the secondary users.

VI. CONCLUSIONS

In this paper we show there could be low cost solutions that scale for a database assisted-DSA system. we present and implement an architecture that enables dynamic spectrum access for any type of network ranging from IoT to cellular. The two main components of this architecture are

Jammer power	Spectrum Analyzer Readings (dBm)	Wired Sensor Readings (dBm)
2 dBm	-77	-60
5 dBm	-40	-25
8 dBm	-38	-22
14 dBm	-30	-18
17 dBm	-27	-11
20 dBm	-25	-5

TABLE I
RSSI MEASUREMENTS FROM SPECTRUM ANALYZER AGAINST WIRED SENSOR FOR TRANSMISSION POWER LEVELS.

the RSSI sensing network, and the DSA server. We use the concept of modular design in these components which allows transparency between them, scalability, and ease of maintenance and upgrade in a plug-n-play manner, without requiring any changes to the other components. Moreover, we provide a blueprint on how to use off-the-shelf commercially available software configurable RF chips to build low cost spectrum sensors. Using testbed experiments, we demonstrate the efficiency of the proposed architecture by comparing its performance to that of a legacy system. We show the benefits in terms of resilience to jamming, channel relinquishment on primary arrival, and best channel determination and allocation. We also show the performance gains in terms of frame error rate and spectral efficiency. Finally, we compare the RSSI sensitivity of our sensors to that of a professional spectrum analyzer.

REFERENCES

- [1] FCC, Third Memorandum Opinion and Order, 2012, pp. 1-35. https://www.whitespacealliance.org/documents/US_TVWS_Rules_FCC-12-36A1.pdf
- [2] Microsoft, "Microsoft Spectrum Observatory Download and Setup Manual", 2014, pp. 1-39. <https://observatory.microsoft.com/>
- [3] International Telecommunications Union, "Essential requirements for a spectrum monitoring system for developing countries", 2011, pp. 1-8. https://www.itu.int/dms_pubrec/itu-r/rec/sm/R-REC-SM.1392-2-201102-I!MSW-E.docx
- [4] FCC, "Technological Advisory Council", June 2014. <http://transition.fcc.gov/bureaus/oet/tac/tacdocs/meeting61014/TACmeeting61014-10-14.pdf>
- [5] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum Sensing for Cognitive Radio: State-of-the-Art and Recent Advances", *IEEE Signal Process. Mag.*, Vol. 29, 2012, pp. 101-116.
- [6] I. Akyildiz, L. Yeol, M. Vuran, and S. Mohanty, Next "Next generation dynamic spectrum access cognitive radio wireless networks: A survey, *Elsevier Computer Networks*, Vol. 62, 2006, pp. 810836.
- [7] S. Debroy, S. Bhattacharjee, and Mainak Chatterjee, "Spectrum Map: Toward predicting the spatial distribution of spectrum usage in CRNs", *ICST CROWNCOM*, 2011, pp. 1-5.
- [8] J. J. Lehtomaki, J. Vartiainen, M. Juntti, and H. Saarnisaari, "Spectrum Sensing with Forward Methods", *IEEE MILCOM*, 2006, pp. 1-7.
- [9] D. Shepard, "A two-dimensional interpolation function for irregularly spaced data", *ACM 23rd National Conference*, 1968, pp. 517-523.
- [10] R. Murty, R. Chandra, T. Moscibroda, and P. Bahl, "SenseLess: A Database-Driven White Spaces Network", *IEEE Trans. on Mobile Computing*, 2012, pp. 189-293.
- [11] T. T. Nguyen, T. M. Nguyen, H. V. Nguyen and K. L. Dang, "Hardware Implementation of Reception Diversity Techniques for Spectrum Sensing Efficiency Enhancement in Cognitive Radio Network", *IEEE WICT*, 2013, pp. 69 - 73.
- [12] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications", *IEEE Commun. Surv. Tutorials*, Vol. II, 2009, pp. 116-130.
- [13] J. Vartiainen, H. Sarvanko, J. Lehtomaki, M. Juntti, and M. Latvaaho "Spectrum Sensing with LAD-based Methods", *IEEE PIMRC*, 2007, pp. 1-5.
- [14] Texas Instruments, "Overview for Proprietary 2.4 GHz", 2016, http://www.ti.com/lids/ti/wireless_connectivity/proprietary_2-4_ghz/overview.page
- [15] HopeRF Transceivers, 2016, http://www.hoperf.com/rf_transceiver/
- [16] Wireless Innovation Forum, "Software Defined Radio", 2002, <http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf>