CityScape: A Metro-area Spectrum Observatory

S. Roy, K. Shin, A. Ashok, D.Aragon
Dept. of Electrical Eng.
U. of Washington,
Seattle, WA 98195
{sroy, ksshin, anish1, daragon}@uw.edu

M. McHenry, G. Vigil
Shared Spectrum Company
1593 Spring Hill Rd
Vienna, VA 22182
{mmchenry, gvigil}@sharedspectrum.com

S. Kannam,
Daintree Technologies
Redmond, WA 98053
shyam.kannam@hotmail.com

Abstract- A collaborative effort between University of Washington, Shared Spectrum Company and Daintree Technologies is resulting in the first metro-scale spectrum observatory – CityScape. In this work, we provide an overview of the system architecture (both hardware and software components), the novel features that distinguish this from others and the design and operational challenges encountered. For further details, refer to [1].

Keywords: Spectrum Observatory, Software defined radio, Cloud storage, Signal analytics

1. INTRODUCTION

The need for more efficient utilization of existing RF spectrum as a means of alleviating the current spectrum crunch relative to burgeoning demand has recently led to development of new spectrum sharing systems based on dynamic spectrum access approaches. In turn, this requires monitoring of spectrum usage [2] over the area of interest to discover the temporally available white spaces (unused channels) at a location, using a set of distributed spectrum monitoring stations. A key enabler of any engineering solution is low cost sensor hardware (such as software defined radios) for data collection in conjunction with appropriate networking and storage resources and a robust software infrastructure that allows for ***persistent and flexible (on-demand) monitoring of spectrum usage.***

In this work, we report on a newly architected (CityScape) spectrum monitoring system https://cityscape.cloudapp.net built on the open source Microsoft Spectrum Observatory (MSO)[1]. At the time of writing, deployment has begun in metro Philadelphia, PA area at partner campus locations who have agreed to host new stations: U. Pennsylvania, Temple U. and Rutgers Camden[2]. While there have been prior spectrum sensor deployments at individual locations sporadically, this is the first *metro-scale* (multiple nodes covering an urban region) observatory with the capability to produce *persistent* (long duration) I-Q measurement data that is archived on the public cloud and made available for post-processing by the wider research community to extract spectrum information of interest.

Fig. 1: Spectrum Observatory Station at UW Campus

The CityScape web interface[3] is a significantly upgraded version of MSO running on a Windows .NET platform and is compatible with the Ettus USRP SDRs (typically, the B2x0 and N2x0) series. Station hardware consists of a wideband antenna on a roof connected to the SDR sensor (Ettus Research USRP N200 + UBX40 daughtercard combination) that is attached to a PC. The PC runs the SDR management software and is remotely accessible as a web client that can be used to configure the scanner process parameters for data acquisition[4]. There are two data acquisition modes:

1. *Raw IQ*: The device is programmed to duty cycle between recording raw I-Q samples ('on time') and spectrogram or short-term FFT ('off time') over the specified frequency range; multiple scans are aggregated over 1-minute duration and stored locally on the PC. The possible values of the duty cycle is determined according to the frequency band, as discussed in Section 3.5.

2. *Time averaged spectrogram (PSD):* The device is programmed to compute average, minimum, maximum Power Spectral Density (PSD) values [3] over the specified frequency range time-averaged over 1-minute intervals. This

[1] MSO was developed in 2013-14 timeframe jointly between Microsoft Research and Azure Last Mile Connectivity Group. Available: http://observatory.microsoftspectrum.com

[2] Current phase of CityScape deployments is expected to be completed by Summer 2017.

[3] cityscape.cloudapp.net

[4] User can choose the scanning parameters via an XML configuration file.

operation is repeated for a 60-minute duration and per-minute time-averaged PSD stored in a file for upload.

In both cases, the file is written to local disk (along with station relevant meta-data) on the PC, then lossless compressed and sent over an Internet backhaul to Azure Public Cloud storage where they are stored for downloading/subsequent processing for display based on user requests[5]. For the raw I-Q and PSD experiments, the available user knobs are explained further in Tables 2 & 3. On the Cloud server, the data is normalized in both time (grouped into 60-minute segments) and frequency (the power spectral density values at associated frequency points on a linear scale are interpolated onto a logarithmic scale with a spacing of 200 points/decade). Post normalization, multiple aggregations – hourly, daily, weekly, and monthly - are calculated on the Azure Cloud along with values for average, minimum, maximum versus frequency for display/download.

2. HARDWARE AND SOFTWARE SUBSYSTEMS

2.1 Sensor Hardware

The CityScape sensor (Fig. 1) consists of various hardware components contained within an environmentally protected Hoffman electrical box along with two accompanying antennas. It is designed to be installed by attaching to a rooftop mast along with the two antennas.

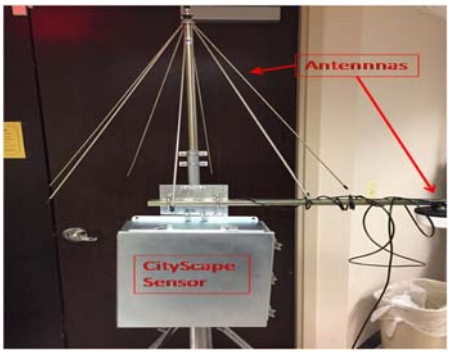

Fig.1: CityScape RF Sensor Enclosure + Wideband Antenna

The primary components within the enclosure are a mini PC, an Ettus USRP, 1 GB Ethernet switch, and a remote power reboot switch. Fig. 2 show the layout of the components within sensor enclosure and a photograph of the actual sensor box that is shipped to hosts.

USRP N200 motherboard in conjunction with UBX-40 daughtercard (RF front end) is recommended as the typical CityScape sensor configuration along with an ASUS mini PC in the sensor box.

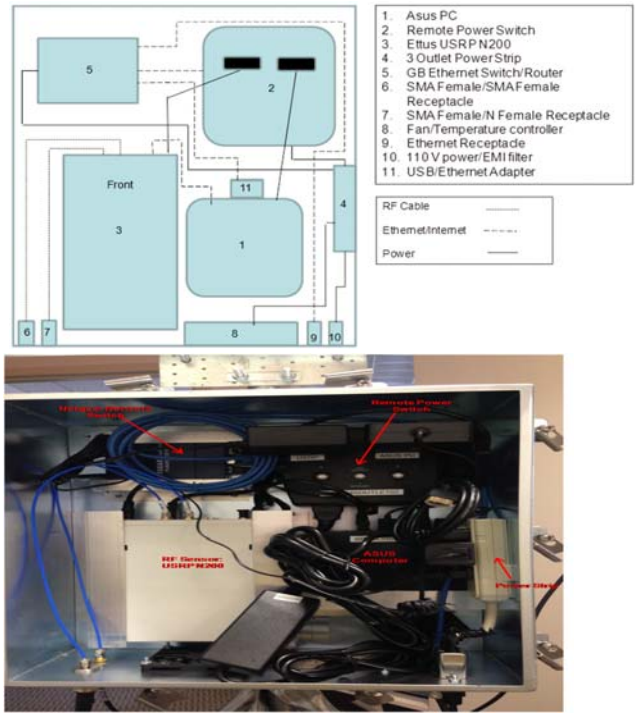

Fig 2: CityScape Sensor Layout and Picture

The miniPC communicates with the USRP via 1000Base-T interface, to command the USRP to tune to desired frequencies and collect data. Block diagrams of the transceiver chains on the N200 and UBX-40 are shown in Figs. 3 & 4, taken from [4].

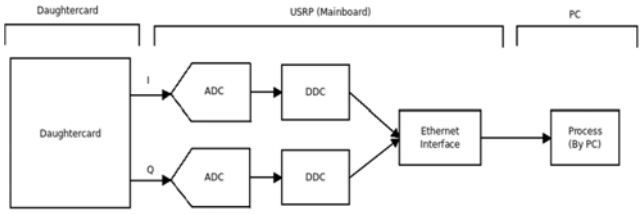

Fig. 3: Block Diagram of USRP

While USRP N200 and UBX-40 are commonly used reputed SDR equipment they are not a drop-in replacement for high quality lab-grade spectrum analyzer.

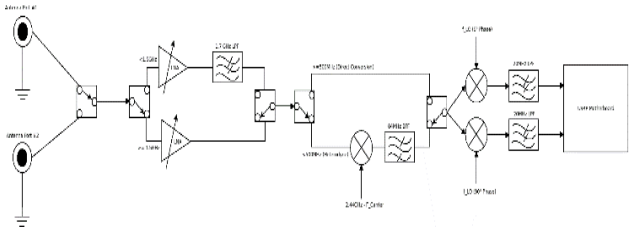

Fig. 4: RX Path of UBX-40 [5].

[5] After upload, the local data files are deleted from the PC.

2.2 *Data Quality Issues*

Some notable issues with our instrument that have come to fore during our deployment include:

- Aliasing, due to the low-pass filters used by the DDC.
- RF Frontend nonlinearity, due to the circuit overload.
- Scan speed.
- Power level calibration.

2.2.1 Aliasing: As shown in Fig. 4, UBX-40 daughtercard has (anti-aliasing) analog filters with bandwidth of 40MHz, that is sufficiently narrower than the I-Q sampling rate of USRP N200 (100MS/s). However, because USRP N200 uses 1000Base-T Ethernet to transport the sampled data to the host PC, the USRP must decimate (sub-sample) this in order to transfer the data to meet Ethernet limits. According to the official documentation [4], 25 MS/s is the maximum effective sampling rate (after decimation) that can be achieved without losing data precision. This is seen by computing the throughput for 25 MS/s sample rate (while the ADC resolution is 14 bits/sample, the data type represents the ADC output with 16 bits; for I-Q sampling, this implies 32 bits/complex value):

$$(25\ MSample/s) * 32\ bits/Sample = 800Mbit/s$$

However, the current decimation on the USRP causes aliasing of the input signal, since the digital low-pass filters used for the decimation process (implemented in the FPGA) do not have requisite band-stop attenuation to prevent aliasing. This issue is further analyzed in Section 3.1.

2.2.2 Power Level Calibration: USRP is used to collect power spectral density estimates of the radio spectrum. However, USRP is not capable of measuring wireless signal power level in dBm. All it can provide to the host PCs are ADC readings (amplitude, in an arbitrary scale). Furthermore, the true receive gain level of the board tend to depend on the frequency that the boards are operating at. So, to compensate this variance and to generate PSD estimates in dBm, a calibration method must be developed and applied to the stations; this is discussed in Section 3.2.

2.2.3 RF Frontend Non-linearities: UBX-40 can operate between 10 MHz to 6 GHz. To achieve this wide tunable frequency range while keeping the cost low, UBX-40 daughtercard employs relatively weak pre-selector filters. It low-pass filters wireless signals with a 1.7GHz low pass filter when it is operating between 10MHz - 1.5GHz, but it is not sufficient in many cases. Due to this, the RF frontend of the sensors - especially the LNAs - can easily fall into the nonlinear regions (covered in Section 3.4), resulting in increased noise figure and spurious signals in the spectrum.

2.2.4 Scan Speed: UBX-40 internally switches its RF frontend circuitries when crossing the 500MHz and the 1.5GHz boundaries. As shown in Figure 2, it acts as a heterodyne transceiver when it is tuned below 500MHz: the received RF signal is up-converted to 2.4GHz and then down-converted to the baseband in the later part of the circuits. When it is operating on or above 500MHz, however, the board acts as a homodyne transceiver which directly converts the received signal to the baseband for sampling. Also, it activates (using switches) a 1.7GHz low pass filter when operating below 1.5GHz. When the USRP undergoes a switch of UBX-40's RF circuits for frequency retune, the board may need some extra time for stabilization. In such cases, the PLL lock flag is not necessarily a good indication of the frequency tuning completion, as the circuit switching & warm-up process often takes longer than the PLL stabilization. This issue is further discussed in Sec. 3.3.

Finally, USRPs may suffer from other imperfections such as the I-Q imbalance (phase/gain imbalance between the I and the Q channel) or the DC Spike (due to the LO leakages, since UBX-40 may operate in homodyne mode). These are well-known issues, and are properly handled by UHD (USRP driver) and the station software.

2.3 *Software*

A high-level overview of CityScape is shown in Fig. 5 and the major software components described in Table 1. The applications are implemented in C# which run on the PC and the Azure cloud.

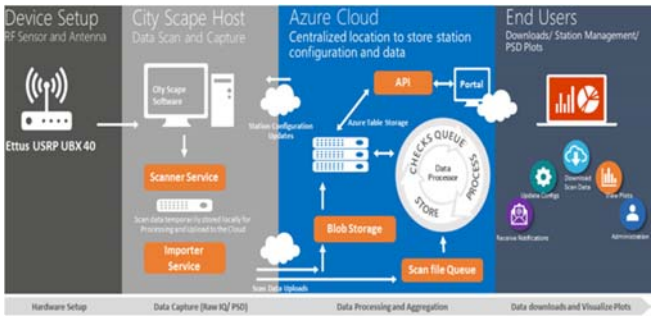

Fig. 5: High level description of CityScape Software

TABLE 1. Major Software Components for Management and Control

Component	Description
ScanFile	The desired PSD and Raw I-Q data, along with the meta-data, i.e. station configurations are stored as PSD and Raw IQ ScanFiles. These files are created on the PC before being uploaded to the Cloud storage.

Scanner Service	Responsible for controlling the RF sensor to perform an experiment and storing data file in a ScanFile format.
Importer Service	Responsible for importing the knob settings to the PC from the web interface. After generating the ScanFiles, it uploads them to the Azure cloud. It also sends a notification when the upload is complete.
Data Processor	Files uploaded by importer service is queued for processing by the data processor attached to the storage account. The processor handles the uploaded ScanFiles - normalization to fixed time/frequency grid and aggregation over various time scales (hours, days, weeks, months) for display in the web portal and stores it in Azure tables.
Azure Storage	All uploaded data for time periods (hourly, Daily, Weekly, and Monthly) are stored in Azure along with associated meta-data for the web portal. Raw I-Q ScanFiles are stored in blob storage within Azure.
City Scape Portal	The processed data is presented to the user in ways that allows viewing and downloading the Azure data across different times and station locations. City Scape Portal enables administrators to manage station access and manage RF sensor configuration remotely.

Multiple Storage Accounts on Azure Cloud

In order to scale in the number of transactions per second and the total amount of storage needed for the spectral data being uploaded, there is a need for multiple storage accounts. Typically, the maximum amount of storage allowed in a single account is 200 TB, and 20,000 transactions per second. Due to the constraint on the storage available on the Cloud, we limit the number of stations per account to 50 measurement stations. After 50 stations, subsequent stations are allocated a new Azure account.

Key Features and Knobs on Web Interface: One of the distinctive features of CityScape is the flexible sensor configuration allowed to end-user for the scan and I-Q data collection. Once submitted through the CityScape portal, the settings are pushed to the Sensor Node where they will configure (or re-configure) the scanning process.

a) Raw IQ Configuration: Configures how I-Q data is captured, as shown in Table 2.

TABLE 2. Raw IQ knobs

Knob	Description
Start/Stop Frequency in MHz	Frequency, inclusive, at which device should start/stop collecting data
On Time	Defines how long data should be captured within a cycle.
Duty Cycle	Fraction of time period during which sensor collects Raw I-Q data.
Retention Seconds	Duration that a Raw IQ data file (.dsor) can be retained on the host PC (time to live), IF it has not been uploaded by the Importer service.

b) Time averaged PSD configuration: The knobs that can be configured for the time averaged PSD experiment are given in Table 3.

TABLE 3. Time averaged PSD knobs

Knob	Description
Minutes of Data Per Upload	Interval (in minutes) for upload of PSD ScanFile (.dsox) to the cloud server.
Seconds of Data Per Sample	Period of time-averaging interval for PSD report. A maximum, minimum and average power spectral density will be calculated for this time period.

c) Scanner configuration: There are two different scan modes for collecting power spectral density: "StandardScan" and "DCSpikeAdaptiveScan". DCSpikeAdaptiveScan tries to remove DC spikes observed from the PSD data obtained and calculated from homodyne receivers (like USRP). Their operation is shown in Figs. 6 and 7. Table 5 shows configuration knobs.

1. Standard scan: Based on the start (f_{start}) and stop (f_{stop}) frequencies, the USRP is tuned to take multiple (n) snapshots, given by

$$n = \frac{(f_{stop} - f_{start})}{BW}$$

where *BW* is the instantaneous snapshot bandwidth. The center frequency of the USRP is retuned sequentially from the lowest to the highest frequency as shown in Fig 7.

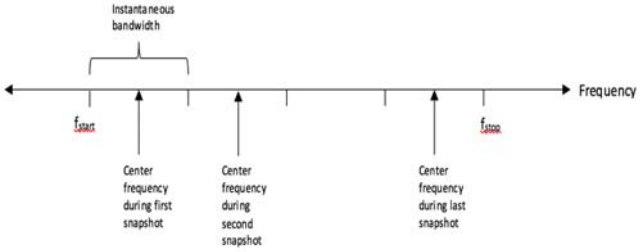

Fig. 7: StandardScan Scan Pattern.

2. DC spike adaptive scan: The spike at DC arises due to the offset in DC voltage or the local oscillator

leakage for the ADC when the RF signal is converted to IQ baseband signal and corrected by DC removal filters [9]. CityScape implements a software technique for DC offset removal as follows. First, a standard scan (discussed above) is implemented followed by another standard scan with a new start frequency given by

$$f'_{start} = f_{start} + \frac{1}{3}BW = f_{start} + f_{offset}$$

where BW is the instantaneous bandwidth. For the power spectral density charts, the FFT of both the scans are computed, and for every snapshot in the first scan, the center third of the FFT values are replaced by the corresponding FFT values from the second scan, as described in Fig 8.

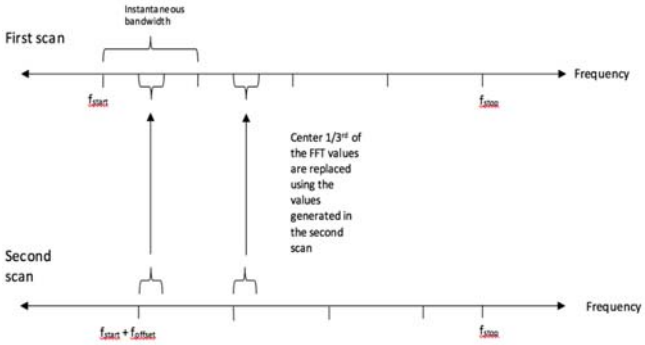

Fig. 8: DCSpikeAdaptiveScan Scan Pattern.

TABLE 3. Scanner configuration knobs

Knob	Description
Start/Stop Frequency in MHz	Frequency(inclusive) at which device starts/stops collecting data
RX Gain	Adjusts the receive gain (dB) of the USRP.
Scan Pattern	StandardScan or DCSpikeAdaptiveScan
Effective Sampling Rate (Hz)	Determines the effective sampling rate (rate after decimation) of the scanner.
PLL Flag Poll Delay	Period of time between each PLL Lock flag polling (which occurs after each frequency retune).
Additional Tune Delay	Duration after each frequency retune (in addition to the PLL flag poll delay).
Samples Per Snapshot	Indicates number of samples to capture for each snapshot.

3. Design and Deployment Challenges

3.1 Aliasing

The analog pre-filter of the UBX-40 daughtercard has much narrower bandwidth (40MHz) than compared to the ADC foldover frequency (100MHz, corresponding to I-Q sampling at 100MS/s). However, due to the bottleneck link between the USRP and the host PC, the I-Q data must be decimated by DDCs (in the FPGA of the USRP motherboard; a block diagram of the DDCs shown in Figure 9) prior to sending to the host PC over Ethernet. The sampling rate of the ADCs is 100MS/s, but even 25MS/s of the effective (decimated) sample rate utilizes 80% of the 1000Base-T link between the PC and the USRP (shown in Section 2). The low-pass filter and decimation process introduces aliasing to the I-Q data as shown in Fig. 10 due to DDCs low-pass filter's (implemented using CIC filters and/or half-band filters) non-ideal stopband. Since the DDC filter design is paired to the decimation rate on USRP, it cannot be directly reconfigured by the user (requires FPGA-level modifications, which is currently out of the scope of this project).

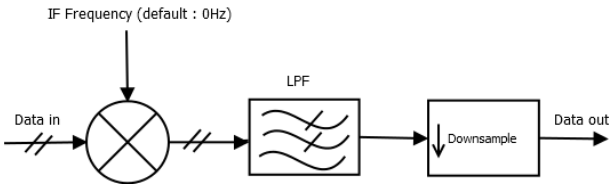

Fig. 9: Block Diagram of the DDC Unit in USRP N200

Fig 10 shows the frequency response of the analog pre-filter of the UBX-40[6].

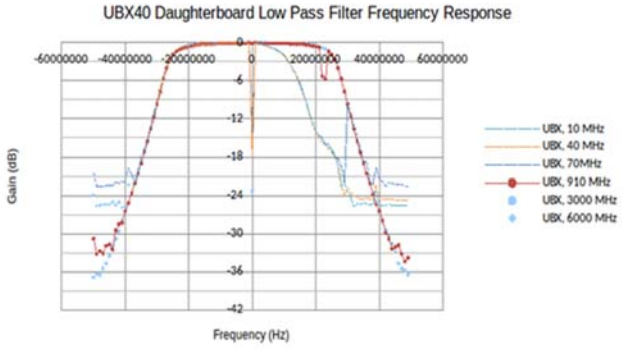

Fig. 10: Analog Filter Characteristics of UBX-40

Fig 11 shows the overall frequency response of the sensor (analog + DDC filter combined) at the effective sampling rate (post-decimation) of 25MS/s. Clearly, signals between -

[6] Some measurement glitches were observed at 10MHz/40MHz center frequencies.

15MHz to -12.5MHz and 12.5MHz to 15MHz can be aliased. For ADC sampling rate of 100MS/s, the attenuation at foldover edge +/-50 MHz is already 35 dB. The combined filter characteristic shown in Figure 11 is obtained as follows:

1. A CW with known power and carrier frequency is injected into the sensor using a loopback cable.

2. The sensor is tuned to the center frequency of the injected CW and I-Q data collected.

3. From the I-Q data, the relative power level of the injected CW is measured.

4. Center frequency of the sensor is changed by a small step and the power level of the (aliased) CW is re-measured.

5. Step 4 is repeated, until the frequency response of the filter is characterized.

The characteristics of the analog filter in Figure 10 is estimated similarly, by changing a 'lo_offset' parameter of the USRP, instead of the local oscillator frequency of the board[7].

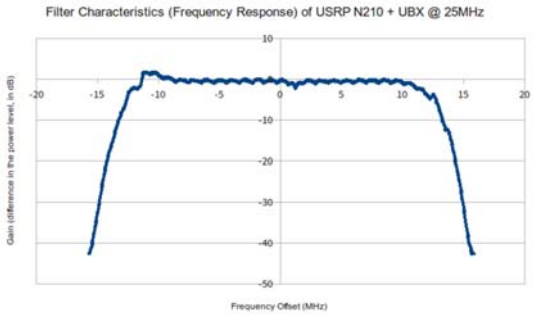

Fig. 11: Filter Characteristics of UBX-40 at 25MS/s

Since the aliasing problem is caused by the DDCs of the USRP, and the low-pass filter used by the DDCs is not user-adjustable, reducing the effective sampling rate (true sampling rate divided by the decimation factor) of the sensor does not address the problem. Furthermore, at some settings (e.g. decimation factor = 5; effective sampling rate = 20 MS/s), the roll-off characteristic of the filter is even worse, due to the inability to use half-band filters when the decimation factor is not even.

Fixes for this problem have been suggested, but not implemented to the station software yet. These include -

- For PSD estimation

a) Discard the edge portion of periodograms where non-negligible aliasing may exist, keeping the center portion and re-scheduling the snapshot frequencies so that all the frequencies in the configured range are still covered.

b) Compensate for the effect of aliasing by analyzing snapshots from neighboring frequencies and post-processing the PSD data. For an example, if the PSD estimate for channels around 875 MHz and around 900 MHz are roughly the same and if the 875 MHz channels lie on the edge portions of the periodograms of snapshots (where the aliasing can occur) but 900MHz channels do not, one can deduce that the observed values at the 875 MHz channels are due to the aliasing and possibly make a compensation.

- For raw I-Q data collection

Better filters (either on the USRP or the host PC) are needed to solve this issue. As already noted, re-design of DDCs implemented on a FPGA to achieve better filter characteristics is currently infeasible. Alternatively, we can implement low-pass DDC filters in software on the host PC. However, filters implemented on the Host PC will yield resulting sensor bandwidth lower than 25MHz, since the maximum transfer rate between the USRP and the PC already limits the effective sampling rate currently to 25MS/s (25MHz bandwidth). Thus the low-pass filter on the host PC must filter the 25MHz bandwidth I-Q data to remove the aliased components.

3.2 USRP Calibration

USRPs used in CityScape stations are calibrated in two ways: with Ettus calibration tools [11], and our own tools. The former is used to suppress the I-Q imbalance of the sensor, while the latter is used to remove the dependence of the measured signal amplitude on the center frequency of the sensor, and to permit dBm measurements from the collected I-Q data.

UHD Calibration Tools: UHD (the interface library of the USRP transceivers) supplies a few command-line calibration tools. They can be used to detect and minimize I-Q imbalance of a specific USRP transceiver, simply by running the calibration commands with antennas detached.

Amplitude Calibration: I-Q data from USRPs are on an arbitrary scale and thus, periodogram values calculated from these I-Q data depends on various device parameters such as gain level of the LNA, type of the RF daughtercard, and the frequency to which the receiver is tuned. Also, the receiver gain in the UBX daughtercard varies with frequency [10]. Amplitude calibration is thus needed to determine the power level in absolute scale (dBm); the I-Q data are calibrated with our custom tools for *each* board via the procedure described below.

1. Using a known signal generator, a CW signal is injected into USRP.

[7] The 'lo_offset' parameter is a special feature of USRP, which allows us to put a low frequency digital IF stage between the analog filter and the decimator of the board.

2. The USRP is tuned to the CW carrier frequency, with a small (100 kHz) offset to separate the CW signal from the DC spike and I-Q data collected.
3. Power level of the CW is estimated (in dB), and compared against the true power level of the signal (in dBm). The result as a function of USRP center frequency is written into a file.
4. The source and USRP frequencies are stepped together to another measurement point.
5. Step 3 is repeated, and the new results appended to the calibration file. The procedure is complete when the maximum carrier frequency that the USRP's daughtercard and the signal generator both support is reached.

During operation, the stations adjust the amplitude of the I-Q data in software using the calibration file, whereby the periodograms of the snapshots will generate PSD estimates in dBm scale. Adjusting the amplitude of the I-Q data is via

$$x_{calibrated} = x \frac{10^{(Cal_{est})/20}}{10^{Gain/20}}$$

where *Gain* is the gain level of the LNA in dB, x is the amplitude of the uncalibrated I-Q data, and Cal_{est} is the calibration factor at the USRP's receive frequency, both in dB. For frequencies not explicitly stored in the calibration file, Cal_{est} is interpolated on a log-frequency scale.

Frequency Calibration: The USRPs do not include the optional (and expensive) GPS-based clock stabilizer, so we had initially anticipated slight differences in USRP base clock frequencies that would require software compensation. However, this concern was not borne out by experience. All of our tested units achieved the desired RF frequencies within a tolerance much narrower than the width of our application's PSD frequency bins. Therefore, no software frequency calibration was implemented.

3.3 Scanning

Since the CityScape sensors continuously re-tune to different center frequencies over wide range, it is important to know when the re-tune process of the USRP completes. Originally, the station software assumed that the re-tune process is complete when the USRP raises a "PLL locked" flag. However, during the development, it has been found that the PLL lock flag of the USRP is not a sufficient indication of the completion of the frequency re-tune process. The sensor often needs a few more milliseconds to be fully stabilized, especially when it crossed the 500 MHz and 1.5 GHz boundaries (where the RF front-end signal path is changed.) Collecting the I-Q data immediately upon PLL lock flag detection caused spurious signals and unexpectedly high noise figure on some bands. To fix this problem, the frequency retune algorithm has been revised as in the following figure.

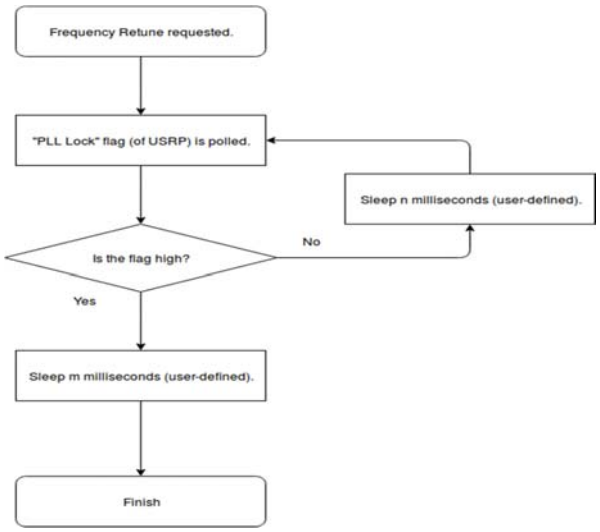

Fig. 12: Frequency re-tune process of the stations.

This modification significantly reduced the amount of spurious signals shown on the PSD plots.

3.4 RF Front-end

The UBX-40 RF daughtercard can be overloaded by strong transmitters in near proximity. The wideband RF front end, essential to our application, implies a weak pre-selector, whereby signals far outside the desired passband will contribute to the total signal strength in the analog hardware. The front end LNA gain is programmable in range from 0 - 31.5 dB (in reality, this sets the gain of an attenuator sandwiched between two fixed LNAs to make one programmable-gain LNA). We found that any setting above 5 dB was sufficient to cause nonlinear behavior at some of our test stations. Power spectral density estimates due to front-end overload tend to show:

1. An unusually high noise floor.
2. Intermodulation products of (out-of-band) signals.
3. Harmonics of (out-of-band) signals.
4. Nonlinear dependence between the LNA gain (programmable) and observed signal power levels.

However, a low LNA gain level increases the noise figure of the board, as this makes the desired signals less distinguishable from the self-noise of the RF circuit. Quantization noise also becomes an issue as weaker signals use fewer bits of the ADC. With insufficient front-end gain, some signals of interest may not be able to flip even the least significant bit. Currently, CityScape sensors do not automatically adjust amplifier gain levels to avoid nonlinearities while maximizing receiver sensitivity. Such

features have been proposed, and will be evaluated in the future. For now, these issues are manually addressed during the station installation process by analyzing the data quality of the PSD estimates (by observing the noise floor and identifying suspected harmonics) and carefully choosing the LNA gain level (fixed).

3.5. Security and Policy Architecture

For a city-scale spectrum monitoring project, data *privacy* is a prime concern. Since many frequency bands are used for transmission of sensitive content for military applications and licensed cellular use carrying private communications, the sensors should not be allowed to continuously store data in these bands. Hence for any given duration, the security architecture should ensure that a sensor can only collect I-Q data intermittently [8]. To implement this, we introduce two knobs: 'ON time' which specifies the (maximum) duration that the sensor can continuously store data at a particular frequency band, and 'Duty-cycle' which specifies the fraction of time in which the sensor can collect I-Q data. It is crucial that the I-Q files are generated in accordance with a policy architecture that is frequency dependent. Here, a duty-cycle approach is employed wherein I-Q files are generated only during the 'ON time' of a time interval, i.e., during the 'OFF time' no I-Q data is generated. Frequency bands are classified into three categories based on the following criteria.

1. *Black:* These frequency bands are used for sensitive Federal applications, and no I-Q data is generated, i.e. duty-cycle, $d_{black} = 0$.
2. *Gray:* These frequency bands are used by cellular technologies, broadcast television and any local unencrypted communications. The I-Q data is generated only during the 'ON time' given by a duty-cycle, d_{gray}.
3. *White:* In these frequency bands, which mainly constitute the unlicensed bands [13], the duty-cycle $d_{white} = 1$, i.e. the experimenter is allowed to generate IQ data continuously in time.

The above classification is region-specific and is determined using the FCC database [12] and the DoD Strategic Spectrum plan [6]. This implies that the security policy settings for stations are different locations could vary.

3.6 Data Representation

Google's Protocol Buffer [7], or Protobuf, data format is used to store the generated data and corresponding meta data. This is done by the Scanner process before applying lossless compression. Protobuf is a language-neutral mechanism to store data, and in CityScape, the metadata fields are designed using C# code. While this is a simple and effective strategy for data storage, but there are a few drawbacks as listed below.

- Protobuf was not designed to store large datasets. Google recommends using it for filesizes smaller than 1 MB but in the CityScape project, large files (varying from 10 MB to 200 MB) can be generated for different settings.
- The maximum file size limit with Protobuf is 512 MB.
- The C# code needs to be updated every time a new software-defined radio is introduced to the project.

A suggested fix to these drawbacks is to use an alternate data representation such as VITA-49 [8] that is readily compatible with open-source GNU Radio module[9]. Further, other drawbacks of Protobuf with respect to the VITA-49 include:

- VITA-49 standard can handle larger filesizes (beyond 512 MB). This allows storing of I-Q data for a longer duration of time per file, which can be beneficial for the experimenter.
- If a new SDR is to be integrated into CityScape, it takes some effort by a developer to ensure data-format compatibility with the existing software. In the case of VITA-49, gnuradio.org regularly updates the tools so the software is compatible and ready for use.

Hence a one-time effort in designing metadata fields and migration to the VITA-49 is required. Once this is done, any updates to the standard is automatically implemented by gnuradio.org and requires no additional effort by the user.

3.7 *Backhaul Constraint*

Each CityScape sensor can generate significant data for upload (as shown in Table 5) via Experiment 1 (duty cycled I-Q collection) to Azure; this naturally creates a burden on the local station host/administrator who has to provision for this Internet backhaul, in terms of data charges and impact on network utilization. Using the recommended settings, we estimate the average upload rate of data from the PC. Using the CityScape file size estimate tool [14], the station administrator can change the available knob settings and compute the average upload rate U as:

$$U = \frac{NL}{\tau} d$$

where the net delay τ is the sum of the processing delay τ_{proc}, re-tune delay τ_{retune} and the data collection delay τ_{col} as shown below.

[8] Note that time-averaged spectrograms do not present such security concerns, as do I-Q data.

[9] The GNU Radio 'file sink' modules implements this.

$$\tau = \tau_{proc} + \tau_{retune} + \tau_{col},$$

N is the number of samples per snapshot, L the size of a sample and d is the duty-cycle factor. τ was measured to be 1.5 ms. For N = 1024 and L = 8 B, Table 5 shows a computation of the upload rate as a function of the duty-cycle factor d.

TABLE 5. Upload rate vs duty-cycle

Duty-cycle d	Upload rate U (MB/s)	Measured Upload rate (MB/s)
1/60	1.36	1.9
2/60	2.73	2.8
1/10	8.19	8.5
1	81.92	97.1

As is intuitive, the duty-cycle can be primarily varied to meet any backhaul constraint on the upload rate of a station. Also, the 'Additional Tune delay' knob can be changed (which increases τ by an amount specified) to lower U.

4. Future Work

A typical issue with our design are limits on the upload bandwidth expected in scenarios with constraints on Internet backhaul. While the upload rate is high, the actual information needed by a user is typically much less (e.g. a short description of the transmitters detected in the data). The challenge is to tune the data analysis to produce the intelligence desired by a user, which is highly variable. Such an analytics toolkit will likely employ a multi-layered architecture for advanced detection/classification to achieve desired sub-noise detection sensitivity, which requires large amounts of processing.

Further, such a utility will also incorporate local control on sensing, scanning and classifier selection to maximize spectrum intelligence while minimizing scan time and processing[10]. The sensing policy engine sets a policy for detection and classification, whether to use a low-cost energy detector or other more reliable but expensive classifier and decides future sensing schedule (what bands are sensed next based on prior results). Such a tool can help minimize Internet backhaul costs and generate alarms (when significant spectrum use changes occur) for quick local action.

Another aspect of our deployment is the cost to install and operate sensors on building roof tops. There are many bureaucratic issues related to liability, safety, limited access and remuneration that increase the installation costs to far exceed the acquisition costs. Installing multiple sensors inside building along windows avoids most of these problems; however each sensor location now has antenna size limits, unknown blockage and resulting coverage limits.

[10] An exemplary system may use a Hidden Markov Model (HMM) based sensing policy engine.

Acknowledgment

This work was funded by NSF CISE Eager Award 1634194 with additional support from Microsoft Azure Last Mile Connectivity Group who have underwritten the cloud storage and processing costs for CityScape effort to date.

References

[1] CityScape Technical Report. Available: https://depts.washington.edu/funlab/wp-content/uploads/2016/08/Cityscape-Technical-report.pdf

[2] S. Haykin, D. J. Thomson, and J. H. Reed. "Spectrum sensing for cognitive radio." Proc. of the IEEE (2009).

[3] S. L. Marple, `Digital spectral analysis: with applications,' Englewood Cliffs, NJ: Prentice-Hall, 1987.

[4] Ettus Research, 'N200/N210', 2017 https://kb.ettus.com/N200/N210.

[5] Ettus Research, 'UBX Daughterboard Data Sheet', 2015 www.ettus.com/content/files/UBX_Data_Sheet.pdf.

[6] Department of Defense (DoD) Strategic Spectrum Plan. https://www.ntia.doc.gov/files/ntia/publications/dod_strategic_spectrum_plan_nov2007.pdf

[7] Google, Protocol buffers, //code.google.com/p/protobuf/

[8] T. Cooklev, R. Normoyle, and D. Clendenen. "The VITA 49 analog RF-digital interface." IEEE Circuits and Systems Magazine 12.4 (2012): 21-32.

[9] R. G. Lyons, "Understanding Digital Signal Processing," 2nd Edition.

[10] RF Characterization Data: UBX daughterboard. https://files.ettus.com/performance_data/ubx/UBX-without-UHD-corrections.pdf

[11] Ettus Device Calibration. Available: https://files.ettus.com/manual/page_calibration.html

[12] FCC Spectrum Dashboard. http://reboot.fcc.gov/reform/systems/spectrum-dashboard

[13] ITU Radio Regulations, Chap. II, Art. 5, Section IV. http://search.itu.int/history/HistoryDigitalCollectionDocLibrary/1.41.48.en.101.pdf.

[14] CityScape Data File Size Estimator, 2017. Available: https://github.com/city-scape/miscellaneous/tree/master/FileSizeEstimate