Spectrum Protection from Micro-Transmissions Using Distributed Spectrum Patrolling

Mallesham Dasari, Muhammad Bershgal Atique,
Arani Bhattacharya, Samir R. Das

Stony Brook University

Abstract. RF spectrum is a limited natural resource under a significant demand and thus must be effectively monitored and protected. Recently, there has been a significant interest in the use of inexpensive commodity-grade spectrum sensors for large-scale RF spectrum monitoring. The spectrum sensors are attached to compute devices for signal processing computation and also network and storage support. However, these compute devices have limited computation power that impacts the sensing performance adversely. Thus, the parameter choices for the best performance must be done carefully taking the hardware limitations into account. In this paper, we demonstrate this using a benchmarking study, where we consider the detection an unauthorized transmitter that transmits intermittently only for very small durations (micro-transmissions). We characterize the impact of device hardware and critical sensing parameters such as sampling rate, integration size and frequency resolution in detecting such transmissions. We find that in our setup we cannot detect more than 45% of such micro-transmissions on these inexpensive spectrum sensors even with the best possible parameter setting. We explore use of multiple sensors and sensor fusion as an effective means to counter this problem.

Keywords: Distributed Spectrum Monitoring, Transmission Detection.

1 Introduction

RF spectrum is a natural resource that is in limited supply but is nevertheless in great demand due to the exponentiating increase of mobile network use. Naturally, just like any such resource the spectrum must be protected against unauthorized use. This issue has recently been exacerbated by the increasing affordability of software-defined radio technologies making RF transmissions of arbitrary waveforms in arbitrary spectrum bands practical.

One way to protect spectrum is via large-scale spectrum monitoring. Various spectrum monitoring efforts have been underway for many years (e.g., [14][13][15]). One issue in such efforts is that lab-grade spectrum sensors are large and expensive both to procure and operate. This issue has recently been addressed by promoting the use of small and inexpensive spectrum sensors that can potentially be crowdsourced, e.g., SpecSense [8], Electrosense [30] and other projects [19][32][6]. This enables much wider deployment in practical settings. Some of these works [8][30] use inexpensive software-defined radios (such as RTL-SDR which costs ≈\$20 [27]) and inexpensive compute devices (such as the RaspberryPi which costs ≈\$40 [25]) attached to these spectrum sensors to enable compute, storage and network capability. ElectroSense has already deployed using these inexpensive spectrum sensors[1] to successfully monitor wide-area spectrum [26].

However, one major concern here is that these inexpensive spectrum sensors are too resource limited and may not be able to perform resource intensive spectrum sensing and detection tasks, e.g., Fast Fourier Transform (FFT) computation and signal detection algorithms [7]. Several sensor related parameters (such as sampling rate, FFT size) and device related parameters (CPU and memory) affect the signal detection performance. A natural question here is to ask how much of these parameters impact the transmitter detection performance given the fact that there are diverse heterogeneous sensors with diverse capabilities. Understanding the impact of these parameters is crucial to design and deploy this class of spectrum sensors.

To address this question, we consider *detecting an intermittent transmitter* as an example problem and characterize the impact of these parameters on detection performance. In our setup, the transmitter here generates a tone of certain duration (e.g., $1\mu s$) periodically. Detecting such 'micro-transmissions' on these inexpensive devices is hard because we cannot use the optimal parameters that are used in general, due to poor compute capabilities. To quantify this, we use detection ratio as a metric to evaluate the system. Detection ratio is the percentage of transmissions detected by the sensor. We characterize the detection ratio for four different parameters: (1) sampling rate, (2) integration size, (3) FFT size, and (4) device hardware. Our goal is to understand how each of these parameters on the inexpensive compute devices affect the detection of micro-transmissions.

Our key finding is that the inexpensive sensors perform very poorly (<45% accuracy) in detecting micro-transmissions on Odroid-C2 board [24] using USRP-B210 [1] and RTL-SDR [27] sensors. In particular, the detection ratio drops by almost 90% for an intermittent transmitter when the transmission duration drops from $1s$ to $1\mu s$ (§2.3). This is because the limited capabilities of the compute devices lead to dropping of samples while computing FFT and power spectral density (PSD). We also observe that increasing sampling rate from 1Msps to 32Msps leads to a drop in the detection ratio by 85% and decreasing it from 1Msps to 512Ksps drops by 30%. This performance impact is due to buffer overflow at the higher sampling rate or insufficient number of samples at lower sampling rate.

We also find that the detection ratio is greatly impacted by device hardware. For example, on Odroid-C2 the detection ratio is 70% and 30% less relative to a desktop PC during local and remote detection, respectively (§3.2). The detection performance can also be impacted by the received power which depends on sensor

[1] We use the term spectrum sensor as sensor and compute device together.

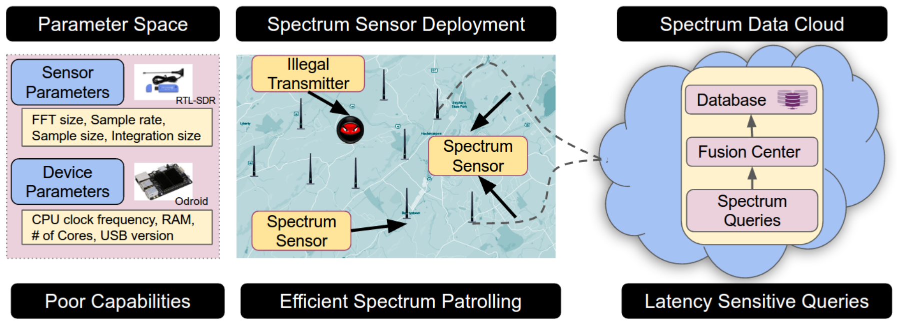


Fig. 1: Architecture of distributed spectrum patrolling. The sensors and attached compute devices are inexpensive and thus suitable for crowd sourcing, but are performance limited.

location, channel conditions as well as the Tx power. To model this behavior at different received power levels, we vary the Tx gain. As expected, the detection ratio reduces significantly on reducing the gain of transmitter (§3.3). Finally to tackle the above challenges in using low-cost spectrum sensors, we deploy multiple sensors in a given location. We show that using multiple sensors and sensor fusion the detection performance improves significantly (§4.2).

2 Distributed Spectrum Patrolling

2.1 Background

The increasing cost of spectrum has made it necessary to monitor its usage and detect illegal spectrum transmissions. Crowdsourcing approaches have been proposed in the past to deploy distributed spectrum sensors at a large-scale. However, that necessitates use of inexpensive sensors. To perform large-scale spectrum monitoring, the most commonly proposed technique is to deploy a distributed set of inexpensive sensors (see in Fig. 1). For example, SpecSense [8] and ElectroSense [30] are two successful, well-known examples of distributed spectrum monitoring. Each such spectrum sensor consists of a low-cost embedded compute platform device such as a RaspberryPi or Odroid-C2 connected to an RF front end such as RTL-SDR or USRP. Each sensor scans the different frequency bands and transfers the sampled IQ data to compute device over USB interface.[2] The compute device either runs signal detection algorithms on the data locally or sends the data to a remote server for processing. We refer to these two configurations as *local* and *remote* processing respectively. A number of such spectrum patrolling systems have been proposed and deployed [8][30][18].

[2] I refers to the in phase component of the signal and Q refers to the quadrature component of the signal. I and Q representation of a signal contains information about the amplitude as well as the phase of the signal. The received IQ samples are used to reconstruct the received signal which is later demodulated to extract the message signal.

Parameters	RTL SDR	USRP B210
Sampling Rate (Msps)	2.5	62
Spectrum (MHz-GHz)	24-1.7	50-6
Bits/Sample	8	12
Interface (USB version)	2	2/3
Cost (≈$)	20	1200

Parameters	Desktop	Odroid C2	RPi3	RPi1
Max Clock (GHz)	3.35	1.5	1.2	0.7
CPU Cores	4	4	4	1
Memory (GB)	8	2	1	0.5
Interface	2/3	2	2	2
Cost (≈$)	1000	40	40	20

Table 1: Spectrum sensor and compute configurations used in our experiments.

A key design challenge for these distributed sensing systems is to decide the type of compute device, the sensor and the associated parameters to use. Devices with better compute power and spectrum sensors with higher sampling rates provide much higher accuracy, but also cost more. The higher sampling rate also increases the network bandwidth requirement which is challenging in a wireless environment. While various deployments use different compute devices and sensors, the performance impact of different device choices is not well understood. To address this question, we systematically benchmark the performance of multiple sensor and compute device parameters in the context of of a specific spectrum patrolling problem where an intermittent transmitter needs to be detected.

2.2 Measurement Setup

Our measurement setup includes the type of spectrum sensors used, the compute devices attached with the sensors, and the data collection process. Each of these are explained below.

Spectrum Sensors: Commodity spectrum sensors vary widely in terms of cost, performance, and the maximum frequency that they can scan. We experiment with two types of sensors — a higher performing but relatively expensive sensor, USRP-B210 (\$1200) [1], and popular, inexpensive sensor, RTL-SDR (\$20) [27]. The RTL-SDR has a maximum sampling rate of 2.5Msps while the maximum of the USRP-B210 is 62Msps. The number of bits per sample is 8 for RTL and 12 for USRP. More bits means better accuracy because of lower quantization noise. The sensor capabilities are summarized in Table 1.

Compute Devices: This device is essentially a small form factor single board embedded computer that acts as a USB host to the sensor. There are many such platforms. We experiment with three different types of devices — Odroid-C2 [24], RaspberryPi-3 (RPi-3) [25] and RaspberryPi-1 (RPi-1) [25], along with a desktop for a basline comparison. Each of these devices vary significantly in terms of cost and performance. The CPU performance directly influences the transmitter detection performance because of the processing needed for the signal detection algorithms. Table 1 summarizes the capabilities of these devices.

Data Collection: For all the experiments, we place the transmitter and sensor at a distance of five meters. This transmitter is a USRP-B210 based software radio that transmits an intermittent tone in the 915 MHz band. The default

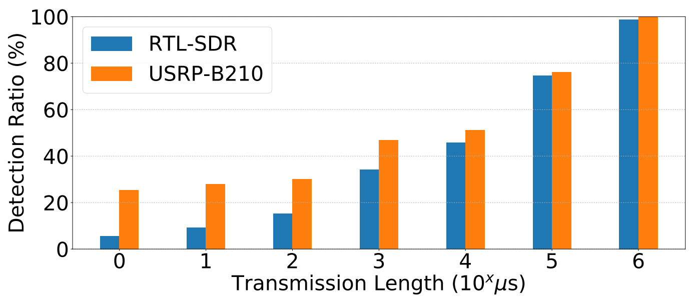


Fig. 2: Detecting intermittent transmissions using USRP-B210 and RTL-SDR sensors on Odroid-C2 board. The detection performance deteriorates significantly as we decrease the transmission length.

transmitter gain for all the experiments is 100. We use an energy-based signal detection algorithm for detecting the transmitter [7]. The algorithm calculates total power within a frequency band by computing FFT on the IQ samples. The signal is detected if the total power in the channel is more than a predetermined threshold. The threshold can be determined by measuring noise in the channel when there is no transmission. For all the experiments, we use 1 ms transmission on Odroid-C2 board with either USRP or RTL sensors. We consider a single transmitter transmitting with a center frequency of 915MHz. The default sampling rate is 1Msps.

2.3 Motivation

As explained in §2.1, these systems employ expensive compute operations on the received signals at the deployed spectrum sensor. For example, most of this prior work performs FFT on the received IQ samples at the sensor itself, computes the PSD and sends the results to a remote server for further analysis. This computation however can slow down the sensor, leading to dropping of IQ samples and thereby making it hard to detect micro-transmissions. To quantify this, we conduct experiments to study the performance of detecting shorter intermittent transmissions by varying the Tx lengths from 1μs to 1s (see Fig. 2).

Based on the length of transmitted signal, a significant difference exists in detection performance even if all other parameters and configurations are identical. While the sensor is able to detect almost all the $1s$ transmissions, the detection performance for $1\mu s$ falls to less than 30% and 10% on USRP and RTL-SDR sensors respectively. This difference must stem from the sensing parameters used on both the sensor and compute device (see §3) as we observe significantly better performance on high-end desktops with the same sensors (not shown here). Note that the desktop machine has sufficient compute power, and we do not expect any drop of samples on it.

Based on this initial study, our goal is to i) understand the factors that influence the performance of detecting micro-transmissions on the inexpensive

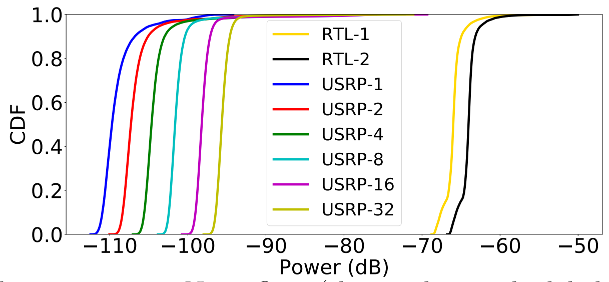


Fig. 3: Sampling rate versus Noise floor (the number in the label indicates sampling rate in Msps). Higher sampling rates bring challenges in detection in terms of noise floor and data rates.

spectrum sensors and ii) explore an alternative to improve the signal detection performance.

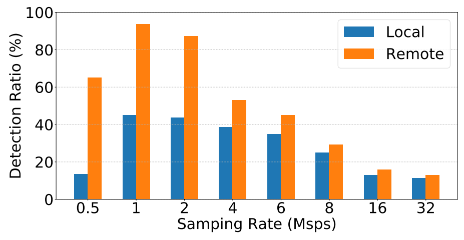
3 Micro-benchmarking of Spectrum Sensors

Four different spectrum sensor parameters (Table 1) could influence the detection performance— sampling rate, integration size, FFT size, and compute device hardware. Sampling rate here is the number of IQ samples received per second. Integration size is the number of samples (i.e., the length of the signal in time) used in a single FFT computation. FFT size is the number of FFT bins. Apart from these parameters, placing the detection locally versus remote, and the transmitter behavior can also influence the detection performance. We study the impact of these properties.

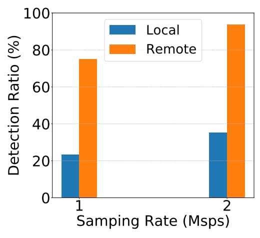
3.1 Sensor Performance

Impact of sampling rate: In general, more the sampling rate, better the transmission detection performance. However, distributed spectrum patrolling with inexpensive sensors brings many challenges in using higher sampling rates: 1) Not all sensors support multiple sampling rates. For example, USRP B210 supports sampling rates from 64Ksps to 62Msps while RTL supports only from 1Msps to 2.4Msps. 2) Higher sampling rates also require proportionately higher backhaul network capacity. 3) Finally, there is a general concern where increasing the sampling rate increases the noise floor which makes it harder to detect micro-transmissions. We study this impact of sampling rate on the inexpensive spectrum sensors.

Fig. 3 shows that the noise-floor increases from -110dB to -90dB when the sampling rate increases from 1Msps to 32Msps on USRP. This becomes much worse, greater than -70dB, for RTL-SDR because of its inaccurate analog converter [5]. This increase in noise-floor makes it hard to choose a threshold for the transmission detection, especially given the fact that there can be many heterogeneous sensors deployed with different sampling rates.



(a) USRP



(b) RTL-SDR

Fig. 4: Detection performance vs. Sampling rate for 1ms transmission. Increase in sampling rate is decreasing the detection performance on USRP-B210.

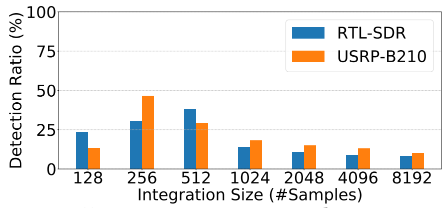


Fig. 5: Integration size vs. detection ratio. We use an FFT bin size of 1024 and run the experiment on Odroid-C2.

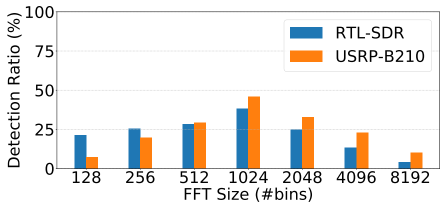
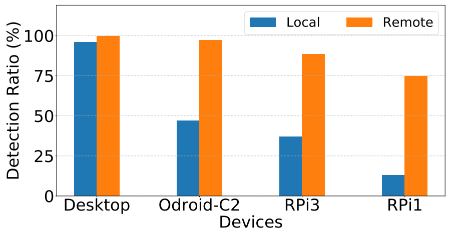


Fig. 6: FFT size vs. detection ratio. We use an integration size of 256 and run the experiment on Odroid-C2.

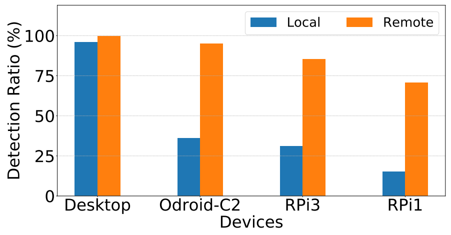
Fig. 4 shows the impact of sampling rate on detection performance. The detection performance using local and remote processing decreases by 30% and 80% respectively from 1Msps to 32Msps sampling rate. This is a counterintuitive result because we expect to see an increase in detection ratio with the increase in sampling rate. This is because the sensor is unable to cope with the speed at which the samples are received under higher sampling rates (remote), and FFT and PSD computation (local). Hence, the sensor is losing many of the important samples that could otherwise detect transmissions. The result is different with RTL-SDR against sampling rate. When sampling rate increases from 1Msps to 2Msps, the detection ratio increases by 15% and 20% for local and remote detection respectively. We explain this by noting that the bits per sample of RTL-SDR is less than USRP, and hence RTL-SDR data require less. Also, reducing the sampling rate below 1Msps decreases detection ratio due to insufficient number of samples.

Impact of integration size: Integration size is a critical parameter in detecting micro-transmissions in terms of both FFT accuracy and compute requirement. Increasing the integration size increases the accuracy of FFT computations, but also increases the amount of computation power needed to compute it. We study the impact of integration size while computing the PSD locally on both USRP and RTL-SDR sensors.

Fig. 5 shows local detection performance against integration size on USRP and RTL-SDR on an Odroid-C2. We set the FFT size at 1024 for this experiment, as we find in the next experiment that it provides the best detection performance. The detection drops by more than 30% from an integration size of 256 to 8192 on



(a) USRP-B210



(b) RTL-SDR

Fig. 7: Impact of device performance on detection performance

USRP. The reason behind this impact is that more number of samples the FFT is computed on, harder it is to detect micro-transmissions as the power is averaged over many noise samples. Another interesting result is that if we decrease the integration time from 256 to 128, the detection rate also drops by over 20%. This discrepancy is because the increased number of FFTs become computationally intensive, and consequently, it is not able to handle all the incoming IQ samples. Similar trend exists with RTL-SDR. Therefore, we observe that the integration size should neither be too low nor be too high.

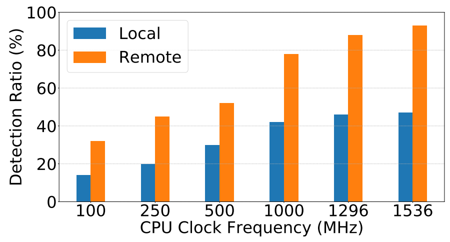
Impact of FFT size: FFT size defines the number of bins while computing the FFT. Each bin represents the resolution of frequency. For example, if sampling rate is 1Msps and FFT size is 1024, then the frequency resolution should be 1MHz/1024 which is 1024Hz. Smaller the frequency resolution (i.e., more bins), more accurately we can detect the power at a given frequency. Also, it increases the amount of computation needed. We evaluate detection ratio with different FFT sizes from 128 to 8192 with local processing on an Odroid-C2. We use an integration size of 256 samples, as we have observed in the previous experiment that it provides the best detection performance.

Fig. 6 shows the impact of FFT size on detection performance. Both RTL and USRP sensors perform better at 1024 FFT size. Having more than 1024 FFT size causes compute and buffer overflow thereby missing IQ samples. Whereas having less than 1024 FFT size makes it hard to detect micro-transmissions. This is because the signal power gets averaged with noise floor due to larger bin size. On the other hand, we find that the optimal FFT size on desktop to detect micro-transmissions is 8192.

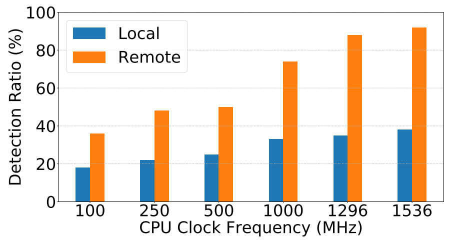
3.2 Device Performance

In the previous section, we studied impact of low-end hardware on the sensor parameters that affect detection performance. In this section, we study the direct influence of device hardware on detection. We keep the best performing sensor parameters such as sampling rate (1Msps) and integration size (256) and evaluate the detection ratio across different devices – Odroid-C2, RPi-3, RPi-1, and a desktop (See Table 1).

Detection performance across devices: Fig. 7 shows the performance of detection for both local and remote processing. We observe that in each case,



(a) USRP-B210



(b) RTL-SDR

Fig. 8: Impact of CPU clock frequency on detection performance

performance of detection reduces with a reduction in the computation power of the computing device. For both USRP and RTL sensors, as we go from more powerful to less powerful computing devices, the local detection ratio drops by over 50% and becomes the worst in case of RPi-1 (<20%). This must be due to the compute capacity of the device as the other parameters are unchanged.

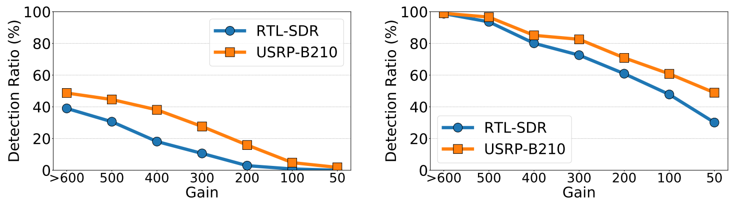
We also find that remote processing has much higher detection ratio than local processing when other sensor and transmitter-related parameters are identical. The compute capacity is a bottleneck for local detection because of the FFT and PSD computation, as we observe that remote detection is as high as 97% on Odroid-C2. Even the remote detection performance degrades to only 60% of total transmissions on RPi-1 as its poor hardware is not able to cope with the sampling rate at which the sensors are sampling.

Critical device bottleneck: To further understand the impact of the device, we experiment with the most critical parameter of the device – CPU clock frequency (as we have seen relatively less impact with other device parameters such as memory and number of cores). A reduction in CPU clock frequency leads to slower computation on the board, and thus lower detection ratio. This is especially important because many such single board compute devices have varied clock frequencies. We conduct the same study for six different clock frequencies available on the Odroid-C2 board. Fig. 8 shows the detection ratio against clock frequency. From 1536MHz to 100MHz clock frequency, the detection ratio for local and remote processing on USRP drops by almost 30% and 62% respectively. Similar trends can also be observed with RTL-SDR. An interesting observation is that the decrease in detection ratio in case of RTL-SDR is less than USRP despite being much cheaper. This is because RTL-SDR has smaller number of bits per sample compared to USRP and hence the compute requirement is less.[3].

3.3 Variation in Transmitter's Behavior

The detection performance depends on the received signal power relative to the noise floor. To model this we vary the transmitter's gain. Gain here is a scaling

[3] Note that RTL-SDR has detection ratio similar USRP when the received signal power is high. RTL-SDR performs poorly when the transmitter gain is very low and signal power is close to noise floor (See §3.3)



(a) Local Processing (b) Remote Processing

Fig. 9: Detection performance vs. transmitter gain. The poor detection performance is because of a combination of factors – dropped IQ samples due to additional computation and poor received signal power level.

factor that decides the power of the transmitted signal. When the received signal power is low (due to the low gain in the Tx for example) there is a significant chance of false alarms. When the signal power is close to the noise floor, it becomes difficult for the sensors to differentiate between noise and the signal. As a result, signal detection becomes difficult because the sensors can falsely tag noise as signal. The false alarm rate increases in this scenario as it is very hard to detect the low power signals unless we keep the threshold close to noise. Keeping the threshold closer to noise increases the probability of false alarm (P_{FA}). We choose a threshold similar to [5] by assuming the P_{FA} as 10% and compute the detection performance based on this threshold.

We experiment with the transmitter changing its gain from 100 to 1000 in steps of 100. Fig. 9 shows the detection performance during local and remote processing. The detection ratio drops to zero when the transmitter changes its gain to 50 during local processing on USRP-B210.[4] The performance is much worse on RTL-SDR, in that it becomes almost zero for gain around 200. This performance degradation also exists for remote processing. The detection ratio drops to 50% and 30% on USRP and RTL-SDR sensors respectively. The reason behind the poorer performance during the local processing is because of the dual impact of PSD computation overheads and lower received signal power levels at lower gain values. During remote processing, only lower gain has an impact on the the detection performance.

4 Discussion

In this section, we discuss the major findings of our study and provide a possible solution to improve the detection performance of the inexpensive sensors.

[4] Note that it is well known that signal power deteriorates as the transmitter decreases its gain. The goal of this experiment is to understand the significance of detecting micro-transmissions under poor capabilities.

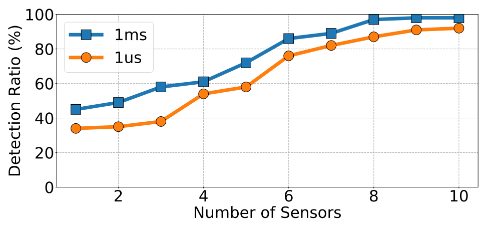


Fig. 10: Improvement in detection performance with number of sensors used.

4.1 Summary of main observations

The key takeaways of our benchmark study are:

- The optimal parameters for spectrum sensing such as effective sampling rate, FFT size, integration size need to be rethought for low-end inexpensive sensors. For example, the optimal sampling rate of detecting $1ms$ length micro-transmissions on a Desktop is 8Msps while on Odroid-C2 it is 1Msps.
- Even when all signal processing is done remotely, the performance impact of using low-end processors in the spectrum sensor could be significant ($< 75\%$ on RPi-1 compared to Desktop, for $1ms$ transmission). This is attributed to two factors– (1) inaccurate spectrum sensors, and (2) poor compute hardware that is not able to process high sampling rates.
- For local processing, availability of compute power is a much bigger factor affecting detection performance than the type of spectrum sensor. However, this is not true for remote processing as the amount of samples dropped in the network during shipping for remote processing is never high enough to reduce detection ratio significantly.

4.2 Data Fusion

In §3, we observed that inexpensive compute devices are limited in terms of computation power because of their hardware. Moreover, the spectrum sensors are also inaccurate in detecting the signal. We now overcome these limitations and improve the signal detection performance while retaining the cost-effective motivation behind distributed sensing. We follow a similar idea from the previous work [8] where the authors show that the inaccuracy of radios can be mitigated by having more radios that are sensing together. This is because the samples are dropped randomly by the devices due to computation bottleneck. The data is later fused from all the sensors. Taking this further, we deploy 10 sensors each with RTL-SDR and Odroid-C2 board at the same location in a campus area.

We use the same transmitter and the setup described in §2.1. The 10 sensors sense the single channel continuously, compute the PSD, and send the power data to our central server. We use Kaa framework [16] and MongoDB [21] to collect the data and store it in a central database. We use a fusing algorithm similar to [5] to combine the data from all the sensors. The detection performance is shown in Fig. 10. We observe that using 8 sensors, a detection ratio of almost 99% and

95% is reached in case of transmission lengths of $1ms$ and $1\mu s$ respectively. This trade-off of cost versus performance benefits shows that detection performance of inexpensive spectrum sensors can be improved by deploying more sensors. A more complicated scenario is to detect the transmissions where the transmitter is changing its gain. This brings the challenges of dealing with P_{FA} while fusing data and requires more sensors to detect all transmissions.

5 Related Work

The advent of inexpensive software radios has made the spectrum vulnerable to unauthorized use [10][17][9][2][31][23]. We discuss two related lines of research: 1) distributed spectrum patrolling, and 2) benchmarking of spectrum sensors.

Distributed spectrum patrolling: Multiple studies such as SpecSense [8], ElectroSense [30] and RadioHound [18] have proposed deploying distributed spectrum patrolling systems using commodity spectrum sensors. However, they all deploy one or two different varieties of sensors and compute devices. For example, RadioHound uses RPi's and laptops as the compute device whereas ElectroSense and SpecSense use RPi's and Odroid-C2's respectively. Other studies such as [4] and [5] have focused on the heterogeneity of the sensors and their impact on detection, or various performance issues related to distributed sensing, such as inaccurate clocks [3] and noisy outputs [22][19]. However, these studies do not investigate impact of sensing parameters or device hardware.

Benchmarking of spectrum sensors: A number of studies benchmark the performance of individual spectrum sensors and the compute devices. For example, [28][7] benchmarks the energy and performance trade-off of RPi and compare it with a smartphone and a laptop based sensor. Other studies investigate the performance of multiple compute devices such as RPi-2, RPi-3 and Beaglebone-Black in the context of audio processing [20][11][12]. Finally, [29] benchmark FFT computations on multiple inexpensive compute devices to study their utility for on-board processing for space missions.

6 Conclusion

The demand for wireless spectrum sharing and co-existence technologies makes large-scale, real-time spectrum measurements necessary. In this work, we explain the key issues that current wide-area distributed spectrum sensing systems face, by benchmarking the impact of sensor and device-related parameters when detecting unauthorized micro-transmissions. We show that the detection performance is no more than 45% even with optimal parameter settings for a $1ms$ transmission. The poor performance is mainly attributed to limited computation capability of the device that results in lost samples. To improve this detection performance, we deploy multiple sensors and demonstrate a 98% of detection performance by fusing the data from all the sensors. We believe that this study also serves the validation and reappraisal of distributed sensing systems such as SpecSense [8] and ElectroSense [26].

Acknowledgments

This work is partially supported by NSF grant CNS-1642965 and a grant from MSIT, Korea under the ICTCCP Program (IITP-2017-R0346-16-1007).

References

1. USRP B210. https://www.ettus.com/product/details/ub210-kit.
2. Juan Andrés Bazerque and Georgios B Giannakis. Distributed spectrum sensing for cognitive radio networks by exploiting sparsity. *IEEE Transactions on Signal Processing*, 58(3):1847–1862, 2010.
3. Roberto Calvo-Palomino, Domenico Giustiniano, Vincent Lenders, and Aymen Fakhreddine. Crowdsourcing spectrum data decoding. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.
4. Roberto Calvo-Palomino, Damian Pfammatter, Domenico Giustiniano, and Vincent Lenders. A low-cost sensor platform for large-scale wideband spectrum monitoring. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, pages 396–397. ACM, 2015.
5. Ayon Chakraborty, Arani Bhattacharya, Snigdha Kamal, Samir R Das, Himanshu Gupta, and Petar M Djuric. Spectrum patrolling with crowdsourced spectrum sensors. In *IEEE INFOCOM*, 2018.
6. Ayon Chakraborty and Samir R Das. Measurement-augmented spectrum databases for white space spectrum. In *CoNEXT*, pages 67–74. ACM, 2014.
7. Ayon Chakraborty, Udit Gupta, and Samir R. Das. Benchmarking resource usage for spectrum sensing on commodity mobile devices. In *Proceedings of the 3rd Workshop on Hot Topics in Wireless*, HotWireless '16, pages 7–11, New York, NY, USA, 2016. ACM.
8. Ayon Chakraborty, Md Shaifur Rahman, Himanshu Gupta, and Samir R Das. Specsense: Crowdsensing for efficient querying of spectrum occupancy. In *INFOCOM*, pages 1–9. IEEE, 2017.
9. Ruiliang Chen, J-M Park, and Kaigui Bian. Robust distributed spectrum sensing in cognitive radio networks. In *INFOCOM*, pages 1876–1884. IEEE, 2008.
10. C Cordeiro, K Challapali, et al. Spectrum agile radios: utilization and sensing architectures. In *DySPAN*, pages 160–169. IEEE, 2005.
11. Mallesham Dasari, Conor Kelton, Javad Nejati, Aruna Balasubramanian, and Samir R Das. Demystifying hardware bottlenecks in mobile web quality of experience. In *Proceedings of the SIGCOMM Posters and Demos*, pages 43–45. ACM, 2017.
12. Mallesham Dasari, Santiago Vargas, Arani Bhattacharya, Aruna Balasubramanian, Samir R Das, and Michael Ferdman. Impact of device performance on mobile internet qoe. In *Proceedings of the Internet Measurement Conference 2018*, pages 1–7. ACM, 2018.
13. NASA RF Propagation Database. https://propagation.grc.nasa.gov/.
14. MTP Group et al. Microsoft spectrum observatory,[online], seattle,[date of reference, nov. 2013.].
15. Anand Iyer, Krishna Chintalapudi, Vishnu Navda, Ramachandran Ramjee, Venkata N Padmanabhan, and Chandra R Murthy. Specnet: Spectrum sensing sans frontieres. In *NSDI*, pages 351–364. USENIX Association, 2011.
16. Kaa. https://www.kaaproject.org/.

17. Mojgan Khaledi, Mehrdad Khaledi, Shamik Sarkar, Sneha Kasera, Neal Patwari, Kurt Derr, and Samuel Ramirez. Simultaneous power-based localization of transmitters for crowdsourced spectrum monitoring. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 235–247. ACM, 2017.
18. Nikolaus Kleber, Abbas Termos, Gonzalo Martinez, John Merritt, Bertrand Hochwald, Jonathan Chisum, Aaron Striegel, and J Nicholas Laneman. Radiohound: A pervasive sensing platform for sub-6 ghz dynamic spectrum monitoring. In *Dynamic Spectrum Access Networks (DySPAN), 2017 IEEE International Symposium on*, pages 1–2. IEEE, 2017.
19. Zhijing Li, Ana Nika, Xinyi Zhang, Yanzi Zhu, Yuanshun Yao, Ben Y Zhao, and Haitao Zheng. Identifying value in crowdsourced wireless signal measurements. In *WWW*, pages 607–616. International World Wide Web Conferences Steering Committee, 2017.
20. Andrew P McPherson, Robert H Jack, Giulio Moro, et al. Action-sound latency: Are our tools fast enough? 2016.
21. MongoDB. https://www.mongodb.com/.
22. Ana Nika, Zhijing Li, Yanzi Zhu, Yibo Zhu, Ben Y Zhao, Xia Zhou, and Haitao Zheng. Empirical validation of commodity spectrum monitoring. In *SenSys*, pages 96–108. ACM, 2016.
23. Ana Nika, Zengbin Zhang, Xia Zhou, Ben Y Zhao, and Haitao Zheng. Towards commoditized real-time spectrum monitoring. In *Proceedings of the 1st ACM workshop on Hot topics in wireless*, pages 25–30. ACM, 2014.
24. ODROID-C2. https://wiki.odroid.com/odroid-c2/odroid-c2.
25. Raspberry Pi. https://www.raspberrypi.org/.
26. Sreeraj Rajendran, Roberto Calvo-Palomino, Markus Fuchs, Bertold Van den Bergh, Héctor Cordobés, Domenico Giustiniano, Sofie Pollin, and Vincent Lenders. Electrosense: Open and big spectrum data. *IEEE Communications Magazine*, 56(1):210–217, 2018.
27. RTL-SDR. https://osmocom.org/projects/rtl-sdr/wiki/rtl-sdr.
28. Ahmed Saeed, Khaled A Harras, Ellen Zegura, and Mostafa Ammar. Local and low-cost white space detection. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 503–516. IEEE, 2017.
29. Benjamin Schwaller. *Investigating, Optimizing, and Emulating Candidate Architectures for On-Board Space Processing.* PhD thesis, University of Pittsburgh, 2018.
30. Bertold Van den Bergh, Domenico Giustiniano, Héctor Cordobés, Markus Fuchs, Roberto Calvo-Palomino, Sofie Pollin, Sreeraj Rajendran, and Vincent Lenders. Electrosense: Crowdsourcing spectrum monitoring. In *DySPAN*, pages 1–2. IEEE, 2017.
31. Tevfik Yucek and Huseyin Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE communications surveys & tutorials*, 11(1):116–130, 2009.
32. Tan Zhang, Ning Leng, and Suman Banerjee. A vehicle based measurement framework for enhancing whitespace spectrum databases. In *Mobicom*, pages 17–28. ACM, 2014.