SpecNet: Spectrum Sensing Sans Frontières

Anand Padmanabha Iyer
Microsoft Research India

Krishna Chintalapudi
Microsoft Research India

Vishnu Navda
Microsoft Research India

Ramachandran Ramjee
Microsoft Research India

Venkata N. Padmanabhan
Microsoft Research India

Chandra R. Murthy
Indian Institute of Science

Abstract

While the under-utilization of licensed spectrum based on measurement studies conducted in a few developed countries has spurred lots of interest in opportunistic spectrum access, there exists no infrastructure today for measuring real-time spectrum occupancy across vast geographical regions. In this paper, we present the design and implementation of SpecNet, a first-of-its-kind platform that allows spectrum analyzers around the world to be networked and efficiently used in a coordinated manner for spectrum measurement as well as implementation and evaluation of distributed sensing applications. We demonstrate the value of SpecNet through three applications: 1) remote spectrum measurement, 2) primary transmitter coverage estimation and 3) SpectrumCop, which quickly identifies and localizes transmitters in a frequency range and geographic region of interest.

1 Introduction

Radio Frequency (RF) spectrum measurement studies [9, 10, 5, 7] have confirmed that vast spans of licensed spectrum, deemed white-spaces, are heavily under-utilized. Such studies have helped make a case for allowing unlicensed devices to utilize unused parts of the spectrum opportunistically. Opportunistic Spectrum Access (OSA) is now increasingly seen as a necessity to meet the growing demands of wireless applications. In fact, the historic FCC ruling in 2008 permitting such opportunistic use (and in 2010 allowing use without the need to sense primaries) is a testament to the success of these measurement studies.

Nevertheless, most spectrum measurement studies to date have been conducted in a few developed nations, using only a handful of spectrum analyzers. Even today, the US remains the only country to have allowed an OSA model. Many more measurement studies, especially in developing nations, are perhaps necessary to make the OSA model accepted worldwide.

Further, these measurements represent static spectrum occupancy information over small parts of a country. While spectrum allocation is mostly static today, the adoption of OSA will result in much more dynamic use of spectrum. Thus, access to real-time spatio-temporal maps is beneficial for OSA devices to sense other OSA devices and determine which parts of the spectrum are free/lightly loaded. However, there exists no infrastructure today for measuring real-time spectrum occupancy across vast geographical regions.

Over the past few years, several researchers have proposed novel schemes for efficient media access and network design in white-spaces [3, 20]. Other researchers have proposed novel collaborative spectrum sensing techniques [11] to allow robust detection of spectrum occupancy. However, thorough evaluation of these techniques using real data is hard today. Further, cross-geographic questions such as "How do spatio-temporal access usage patterns in India differ from those in the US?" or "How would a certain OSA technique that works well in the US perform in the UK?" cannot be answered today.

The primary contribution of this paper is SpecNet—a platform that allows researchers across the world not only to conduct spectrum measurement studies remotely in real time, but also implement and test novel distributed collaborative spectrum sensing applications for OSA. SpecNet advances OSA in several ways. First, it helps gather spectrum data in many countries, thereby helping the adoption of the OSA model worldwide. Second, by providing real-time spectrum occupancy maps, OSA devices may be able to quickly identify lightly loaded parts of the spectrum. Third, it provides real trace data that can be used to evaluate novel research ideas in OSA. Finally, in countries such as India, where there is no readily available database of primary users, it can help create an accurate database that can be used by OSA devices.

In SpecNet (Section 4), participant owners of spectrum analyzers register and connect their instruments to the SpecNet server. Each owner volunteers to provide time periods when SpecNet users are allowed to use the instrument to remotely conduct experiments. SpecNet provides its users with a rich API implemented as XML-RPC calls. Thus, SpecNet users can develop and remotely execute measurements or distributed sensing applications in a programming/scripting language of their choice. To the best of our knowledge SpecNet is the first programmable distributed spectrum sensing platform of its kind. SpecNet can be accessed at [15].

SpecNet provides an API that supports three classes of users (Section 4.2). For sophisticated users, SpecNet provides full access to the low-level APIs of the spectrum analyzer. For policy users and others mainly interested in measurement data, say for longituidinal analysis, SpecNet provides APIs that allow access to historic measurement data that SpecNet collects and stores in a database. For other users such as network operators or government personnel, SpecNet provides a set of high-

level APIs that allow these users to write novel applications without having to worry about the intricacies of the spectrum analyzer. For example, a government user interested in spectrum occupancy data need only specify the part of the spectrum (*e.g.*, 500-800 MHz), the geographical boundary (*e.g.*, specified by a center and radius of a circular region), the time interval (*e.g.*, between 12:00 - 16:00 hrs today) and the minimum signal strength of the transmitter that needs to be detected (say -95 dBm). Behind the scenes, SpecNet determines the group of relevant spectrum analyzers and their respective settings that will help satisfy the measurement request, executes the task on these spectrum analyzers and delivers the results to the user. Other users such as OSA network operators may be interested in determining the coverage of their networks at locations where spectrum analyzers may not be available. SpecNet provides an interpolation tool that uses measurements from nearby spectrum analyzers to estimate power at the location(s) of interest.

Given that spectrum analyzers are expensive ($10-40K) and their time of availability for SpecNet's use might be restricted depending on the owner's needs, an important design goal for SpecNet is efficient management of spectrum analyzer time. When two or more spectrum analyzers lie in the region of interest, it may be possible to coordinate their measurements in a manner so as to reduce the overall scanning time while satisfying the user's request. One approach could be to partition the frequency spectrum equally among all the spectrum analyzers in the region of interest. Another approach is to leverage the spatial diversity in the locations of the spectrum analyzers and partition the scanning efforts geographically. Finally, a hybrid approach that combines these two approaches is also feasible.

Two fundamental tradeoffs underlying the very physics of spectrum measurements make this problem of partitioning the measurement task among spectrum analyzers a significant challenge. First, the *time-frequency uncertainty principle* dictates that the finer the resolution of the spectrum scan, the longer it takes to perform the scan. Second, weaker signals require longer scan times to be amenable to detection. Further, the heterogeneity in capability as well as processing speeds across different models of spectrum analyzers adds to the complexity. SpecNet considers these tradeoffs and uses a novel task partitioning scheme for scheduling individual spectrum analyzers (Section 5).

We demonstrate the power of SpecNet through three applications (Section 7). The first application is simply a spectrum scan that is performed across different countries, illustrating the ability to conduct remote measurements. The second application is a coverage estimation application that may be useful to network operators. The application first helps localize a TV transmission tower and then predict its footprint so that operators may avoid the primary owner of the spectrum. This is especially useful in developing countries where a database of primary transmitters is unavailable or incorrect. The third application is SpectrumCop, which may be of interest to government users. Today, it is hard to detect violators of spectrum policy unless a primary owner of the spectrum complains of interference. The SpectrumCop application allows a user to quickly detect and localize a transmitter in a given frequency range and geographic region, demonstrating the utility of SpecNet's coordinated sensing platform.

Thus, we make the following contributions:

- We present the design and implementation of a novel platform called SpecNet that allows spectrum analyzers around the world to be networked and used in a coordinated manner for remote measurement as well as testing and implementation of distributed sensing applications. SpecNet is open for access at [15].
- We present a scheduling algorithm for coordinating measurements among neighboring spectrum analyzers that optimizes spectrum analyzer usage time.
- Finally, we present three applications that demonstrate the value of the SpecNet platform.

2 Related Work

Measurement Studies. One of the earliest studies that aimed at quantifying spectrum usage [9] is by the Shared Spectrum Company. The study, conducted at six different locations in the US, concluded that the average occupancy of spectrum was about 5.2% in the 30 MHz to 3 GHz frequency range. A study by McHenry *et al.* [10] in Chicago and New York revealed that the occupancy was limited to 17% and 13% respectively. Since then, there has been a number of measurement studies [5, 7, 19] in different parts of the world. The common finding of all these studies has been that spectrum is heavily underutilized. In [4], authors derive various statistics from the collected data, and propose a prediction algorithm for channel availability.

All of these studies have been performed using a handful (maximum of 4 according to [4]) of spectrum analyzers scanning spectrum in a small geographical region in an uncoordinated fashion. In contrast, SpecNet provides a platform for coordinating spectrum analyzers across different geographical regions, thus opening doors to more interesting measurement studies. Further, it also enables building occupancy maps of large geographical areas over long durations for longitudinal analysis.

Whitespace Research. Whitespace networking has been gaining attention as an important research field in the networking community. In [3], the authors propose

a Wi-Fi like system built on UHF whitespaces. Yang *et al.* propose a distributed spectrum access technique using frequency agile radios transmitting in orthogonal frequencies [20]. Most of these proposals have been evaluated in restricted settings. We believe that SpecNet would aid whitespace research by allowing evaluation of proposals based on broader, more real-world data. For instance, spectrum measurement data from different continents could be used to evaluate detection techniques.

Cooperative Sensing & Sensor Networks. Cooperative sensing is a well explored topic [11, 16, 6]. The main focus of these papers is detecting a primary whose frequency of transmission and/or location is known. Moreover, the emphasis is on novel collaborative detection techniques. SpecNet and research in collaborative sensing are complementary to each other. For example, measurements from SpecNet can be useful for evaluating these collaborative detection algorithms while advanced collaborative detection techniques can be incorporated into the SpecNet platform as an API.

SpecNet uses Voronoi partitioning for optimizing scan time of spectrum analyzers. The use of Voronoi diagrams has been proposed in sensor networks as well [17, 2]. However, the main motivation for applying a partitioning scheme in sensor networks has been energy savings and/or interference avoidance. Thus, the problem formulations and objective functions are very different.

Testbeds/Platforms. A number of distributed research testbeds/platforms have been built by the community [12, 1, 18]. To the best of our knowledge, SpecNet is the first platform targeted at co-ordinating spectrum analyzers across geographical regions.

3 Spectrum Sensing Using Spectrum Analyzers - A Primer

In this section we attempt to answer the question,"what are the key settings and choices available to a spectrum analyzer user for spectrum scanning and how do they influence the spectrum sensing process?"

3.1 Spectrum Scanning - An Example

We begin with an example spectrum scan of an active wireless microphone depicted in Figure 1. When scanning using a spectrum analyzer, a user typically needs to specify two key parameters—the *scanning frequency range* and the *resolution bandwidth*. The frequency range, (f_{min}, f_{max}) in MHz, specifies that the user is interested in scanning the spectrum from f_{min} MHz to f_{max} MHz. In Figure 1, the scanning frequency range is $(702.05 \text{ MHz}, 702.35 \text{ MHz})$. Resolution bandwidth specifies the granularity in Hz at which the scan is to be performed—the lower the resolution bandwidth, the greater the observed detail in the scan.

Figure 1 depicts the results of the scan at four different resolution bandwidths. When the resolution bandwidth is 1 MHz, the microphone's transmission is not at all perceivable. Upon reducing the resolution bandwidth to 30 KHz, a single clear peak emerges indicating the microphone's transmission. Further reducing the resolution bandwidth to 10 KHz reveals even finer detail—three distinct peaks, which are the signature of an FM-modulated transmission. At 1 KHz resolution bandwidth, the three peaks are revealed as distinct sharp tones.

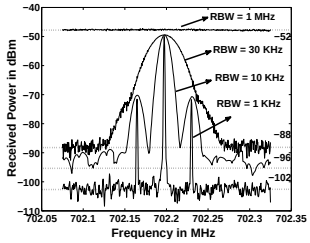


Figure 1: Effect of resolution bandwidth

As seen in Figure 1, a lower resolution bandwidth has two significant effects on the scan. First, greater detail about the signal structure is revealed and second, the noise floor is reduced (from -52 to -102 dBm).

3.2 Occupancy Detection

Often, the goal behind scanning the spectrum is *occupancy detection, i.e.,* to determine which parts of the spectrum have ongoing transmissions. Fundamentally, the problem of occupancy detection attempts to distinguish between signal and noise. While there are several varieties of occupancy detection schemes, perhaps the simplest scheme is to check whether the Signal to Noise Ratio (SNR) is greater than a certain threshold.

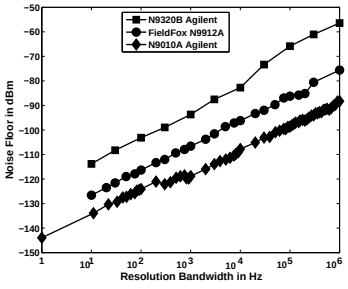


Figure 2: Noise floor versus resolution bandwidth

Dependence of noise floor on resolution bandwidth: As we saw in Figure 1, the noise floor depends on the resolution bandwidth of the scan. This decrease in noise floor arises from the fact that as frequency bins become finer, they accumulate less noise. *A lower noise floor typically results in a greater SNR and consequently more reliable occupancy detection.*

The noise floor (in watts) as a function of resolution bandwidth is typically given by

$$N \propto \rho \tag{1}$$

In Eqn 1, the proportionality constant depends on the spectrum analyzer model, the antenna, the cabling, *etc*. Figure 2 depicts the dependence of noise floor on resolution bandwidth for three different models of spectrum analyzer. While practical measurements indicate minor deviations in linearity, as seen in Figure 2, the linear model (Eqn 1) holds approximately true for all spectrum analyzers we used.

Dependence of detection range on resolution bandwidth: Typically, the farther a transmitter is from a spectrum analyzer, the lower the received power at the spectrum analyzer. The weaker the received signal, the lower the SNR and hence the less reliable its detection. *Detection range* of a spectrum analyzer at a certain resolution bandwidth is the farthest distance from which an ongoing transmission can be detected reliably.

Path loss models such as the Log Distance Path Loss (LDPL) model are typically used to estimate received power as a function of distance. The received power P_r at a distance d from a transmitter transmitting with power P_0 based on the LPDL model is given by

$$P_r = P_0 - 10\gamma \log(d) + L \tag{2}$$

In Eqn 2, γ (usually between 2 and 3 for outdoor environments) is the path loss exponent and L dB (usually modeled as a Gaussian with standard deviation between 5-10 dB for outdoor environments) is a random variable that captures variations in the signal due to fading effects.

If Δ is the minimum SNR required for reliable occupancy detection using a certain detection scheme, then in order to detect a transmission from a distance d, the noise floor must be Δ dB less than P_r, *i.e.*, $P_0 - 10\gamma \log(d) - \Delta$. Since noise floor is dictated by the resolution bandwidth (Eqn 1), this in turn implies that *one must choose a lower resolution bandwidth to reliably detect a transmitter that is farther away from the spectrum analyzer.* The dependence of detection range d on resolution bandwidth can be derived from (Eqn 1) (after converting from dB) as

$$\rho \propto \left(10^{\frac{P_0-\Delta}{10}}\right) d^{-\gamma} \tag{3}$$

Eqn 3 indicates an important aspect of detecting transmissions from a distance, namely, *the maximum usable resolution bandwidth decreases super-linearly (as d^{γ}) with detection range.*

4 SpecNet Architecture

SpecNet is a shared infrastructure consisting of geo-distributed, networked, programmable spectrum analyzers that are contributed and used by the community. The following two goals drive the design of SpecNet. *1) Ease of Use:* We expect SpecNet to support the needs of three different classes of users. First, sophisticated users such as whitespace researchers will likely need real-time, low-level access to the full functionality of the spectrum analyzers. Second, some users such as spectrum policy researchers may simply need access to the data collected by the spectrum analyzers. Finally, users such as secondary network service providers or government personnel interested in spectrum monitoring may require high-level APIs that abstract the details/complexity of SpecNet and provide services such as tower localization or spectrum occupancy detection. *2) Efficiency:* Given that spectrum analyzers are expensive ($10-40K) and may be available to SpectNet for limited duration, it is important that the usage of spectrum analyzers be optimized where possible. Since the spectrum analyzers cannot be arbitrarily "time-sliced" for fine-grained sharing, optimization requires completing each task as efficiently as possible. We now present an overview of the SpecNet architecture.

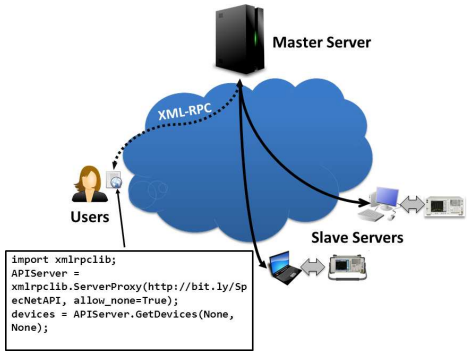


Figure 3: SpecNet Architecture

4.1 Overview

The SpecNet architecture is shown in Figure 3. It contains three key components: users or clients, slave servers that comprise laptops/PCs connected to spectrum analyzers, and master servers that manage the slave servers. The typical work-flow is as follows: clients submit jobs to the master servers; the master servers translate these jobs into spectrum analyzer commands based on Standard Commands for Programmable Instruments (SCPI) [14]. The master server also schedules these at the appropriate slave server nodes for execution at the desired/available time. The output of the commands is then either forwarded immediately to the client or the client is notified of when/where the output data from the submitted job would be available.

XML-RPC: In order to support a wide range of client platforms, the SpecNet service is exposed by the master servers as XML-RPC calls, *i.e.*, remote procedure calls that are encoded in XML and transported over HTTP using the XML-RPC standard. This allows clients to

post jobs using the SpecNet APIs from any Internet-connected node, written in any language of their choice.

Push-vs-Pull: The jobs posted to the master server can either be pushed to or pulled by the slave servers. While a pull-based publish-subscribe model is less complex in terms of state maintenance at the server, it is not suitable for SpecNet users who may want to execute jobs with inter-dependent API calls that require reaction at sub-second intervals (see the Spectrum Cop application in Section 7.3). We thus adopt a push-based model where a persistent TCP connection is maintained between the slave servers and the master servers and jobs are pushed to the slave servers.

Registration: Users contributing slave servers need to first register with the SpecNet master server. They may specify times during which the nodes are available to SpecNet. Upon completion of registration, a simple daemon is downloaded and executes on the slave server. This software establishes an outbound persistent TCP connection to the master server and another connection to the spectrum analyzer, thereby serving as a bridge between the master server and the spectrum analyzer.

Benchmarking: The master server first runs a suite of experiments to benchmark the fundamental characteristics such as noise floor and scan times of each spectrum analyzer (details in [8]). This benchmarking helps the master server efficiently schedule jobs at the slave server nodes. Further, this is also necessary for abstracting some of the low-level details of the spectrum analyzer through higher-level APIs, necessary for masking some of the heterogeneity among spectrum analyzers. We discuss this next.

4.2 APIs

As mentioned earlier, SpecNet is designed to support three classes of users. Table 4.2 lists a subset of the APIs supported by SpecNet.

For sophisticated users who require low-level access to the spectrum analyzer, SpecNet has a reservation API that users can use to reserve a block of time on the desired slave servers. The users can then issue their desired low-level commands, which are simply forwarded through the master server to the slave servers for execution.

For policy users and others who are interested mainly in spectrum usage data, possibly for longitudinal studies, SpecNet schedules up to 10% of the available time at each slave server for itself. During this time, the server performs a high resolution scan of the entire spectrum, stores this data in a SQL database and exposes this data to users through APIs such as `GetPowerSpectrumHistory()` or `GetOccupancyHistory()`. This stored data can also serve as a cache and may help respond (partly) to other submitted jobs.

The interesting challenges in SpecNet's design arise mainly in supporting the third class of users (*e.g.*, network operators). These users may require support for high-level APIs that abstract out many of the details of using spectrum analyzers. While we have designed a few of these APIs (6-9 in Table 4.2), we expect the set of high-level APIs to expand over time based on interest and through community contributions.

Localization and Interpolation: Estimating the geographical coverage of a primary transmitter is essential to creating a spectrum usage map. However, this requires knowledge of specifics of the transmitter such as its location and transmit power. Such information is usually not available or may be incorrect, especially in developing countries (Section 7.2).

In order to localize transmitters, SpecNet provides the `LocalizeTransmitter()` API that uses signal strength observed at spectrum analyzers from various locations but does not require input of parameters such as location and transmit power of the transmitter. Instead, SpecNet estimates these parameters that best explain the signal observations (in least mean square error terms) using well known path loss models such as Longley-Rice or Log Distance Path Loss (LDPL). The number of unknowns that can be estimated, however, fundamentally depends on the number of different locations from which signal strength was observed. In case of the LDPL model (Eqn 2), for example, if signal strengths from only three locations are available, SpecNet sets $\gamma = 3$, takes the transmit power (P_0) as input from the user and estimates the location through triangulation. If signal strength from four different locations are available, SpecNet can estimate P_0 and the transmitter location simultaneously by choosing $\gamma = 3$. When observations from five or more locations are available, SpecNet can estimate the transmitter location, transmit power P_0 and γ simultaneously that best fit the observations. Once the location of the transmitter and other parameters are determined, constructing a spectrum map is straightforward. SpecNet provides the `FindPowerAtLocation()` API that takes these parameters and predicts the likely received power at desired new locations (*e.g.*, locations with no spectrum analyzer).

Spectrum Occupancy Detection: The next two high-level APIs help users obtain spectrum occupancy at desired locations. The `GetPowerSpectrum()` is simply a spectrum scan over a given frequency range on a given device, except that users do not even need to specify the resolution bandwidth. Instead users can specify a region and desired minimum power level of transmitter to be detected. SpecNet then automatically chooses the best resolution bandwidth (based on the fundamental properties of occupancy detection discussed in Section 3)

#	API	Description
	Low-level APIs (*e.g.*, for sophisticated users)	
1	`GetDevices([Boundary], [Timespan])`	Returns a list of spectrum analyzer IDs. Fewer/no arguments possible.
2	`ReserveDevice(ID, Timespan)`	Reserves and returns success, if available.
3	`RunCommandOnDevice(ID,Command)`	Issues SCPI command to device and returns result.
	Commands to access stored data (*e.g.*, for policy users)	
4	`GetPowerSpectrumHistory(ID, Fs, Fe, Timespan)`	Returns (avg) power values from device for given time/frequency range (Fs-Fe).
5	`GetOccupancyHistory(ID/Boundary, Fs, Fe, Timespan, Threshold)`	Returns 0-1 list indicating occupancy in Fs-Fe at device or in region, based on threshold.
	High-level APIs (*e.g.*, for operators or government users)	
6	`LocalizeTransmitter(Boundary, Locations, Powers, Model, Parameters)`	Localizes transmitter inside area, given observed power level(s) at location(s) using Model (LDPL, HATA, Longley-Rice, etc.) .
7	`FindPowerAtLocation(Location, [Transmitter Parameters], Model, [Model Parameters])`	Interpolates power at new location given transmitter location/parameters and model; useful for estimating coverage of transmitter.
8	`GetPowerSpectrum(ID, Fs, Fe, [Boundary, P])`	Schedules a scan for given frequency range (SpecNet determines optimal resolution bandwidth) in order to detect minimum power level P in given area.
9	`GetOccupancy(ID/Boundary, Fs, Fe, P)`	Provides a 0-1 list corresponding to frequencies occupied at a device or region. P is the minimum transmitter power (SpecNet minimizes scan time).

Table 1: Core APIs supported by SpecNet

and returns the results. `GetOccupancy()` API goes further by allowing the user to specify a region of interest for detecting occupancy of signals above a given threshold, without even identifying the desired slave server IDs. This API is useful for applications like Spectrum Cop (Section 7.3), which monitor unauthorized spectrum usage. To support this API, SpecNet computes the optimal set of spectrum analzyers and their corresponding resolution bandwidth values that minimize scan time and returns the results. Optimizing scan time across multiple spectrum analyzers is a challenging problem which we discuss next.

5 Task Scheduling in SpecNet

SpecNet allows users to deploy and execute spectrum sensing applications in real time. Users expect their sensing tasks to be dispatched and completed as soon as possible. Consequently, SpecNet schedules participant spectrum analyzers in a manner so as to minimize task completion time. In this section we describe the challenges posed in the design of a task scheduler for SpecNet.

5.1 Scanning Time of a Spectrum Analyzer

For a spectrum analyzer, the time to perform a scan from f_{min} MHz to f_{max} MHz depends on two parameters namely, *span* $Q = f_{max} - f_{min}$ and the resolution bandwidth ρ used for the scan. Increasing the span requires a spectrum analyzer to scan a larger part of the spectrum and consequently requires a longer scan time. Scanning at a smaller resolution bandwidth requires a larger number of samples to be collected in order to reliably estimate the power in each of the finer frequency bins and hence, more time. For modern spectrum analyzers, the scan time may be modeled as

$$T \propto \frac{Q}{\rho} \tag{4}$$

In Eqn 4, T is the *scanning time*. The proportionality constant in Equation 4 can vary significantly across different models of spectrum analyzers as discussed next.

Theory versus Reality : Figure 4 depicts the scan times measured from different spectrum analyzers at different resolution bandwidths as a function of span. As seen from Figure 4, the dependence of scanning time on span Q is strictly linear as dictated by Eqn 4. Consequently, it is convenient to characterize scan times of spectrum analyzers in terms of *scan time per MHz*, τ. The scanning time for a scan from f_{min} to f_{max} is then determined by the product $(f_{max} - f_{min})\tau$.

Figure 5 depicts the measured scan times per MHz (τ) as a function of resolution bandwidth for three different models of spectrum analyzers in a log-log plot. Based on Eqn 4, the variation of scan times with resolution bandwidth should be linear. However, Figure 5 indicates *significant departure from linearity.* Rather the variation is *piece-wise linear*. For example, for FieldFox N9912A, the variation is linear in sections A-B and C-D separately. The piece-wise linearity arises because spectrum analyzers likely use different sets of circuits and modes for different ranges of resolution bandwidths and these circuits/modes presumably have different performance characteristics. To allow for these non-linearities, SpecNet maintains lookup tables $\tau(\rho)$ describing the scanning time per MHz for a given resolution bandwidth setting for each spectrum analyzer.

5.1.1 Minimizing Scan Time by Automatic Resolution Bandwidth Selection

When scanning a part of the spectrum, users often care about having a low noise floor. The noise floor, however, as discussed in Section 3, depends on the resolution bandwidth chosen. SpecNet allows users to request a scan by a remote spectrum analyzer by specifying the maximum tolerable noise floor. Behind the scenes, SpecNet determines the resolution bandwidth that provides for the fastest scan time that satisfies the required noise floor. In order to enable such an API, the SpecNet server maintains lookup tables that provide scanning times per MHz at various resolution bandwidths, for each Spectrum Analyzer connected to SpecNet.

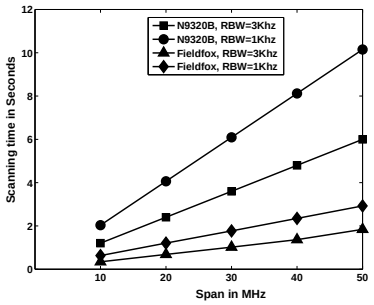


Figure 4: Scanning time versus span

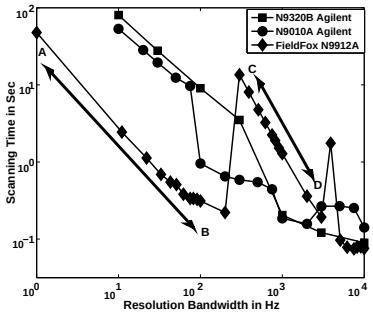


Figure 5: Scanning time per MHz versus resolution bandwidth

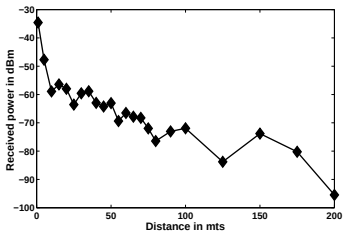


Figure 6: Decay in received signal strength for microphone

Dependence of Scanning Time on Detection Range: A greater detection range requires using a narrower resolution bandwidth (Section 3). This in turn implies that *to increase the detection range of a spectrum analyzer one must accept a longer scanning time.* More specifically, from Equations 3 and 4, scanning time depends on detection range as

$$T \propto \left(10^{-\frac{P_0-\Delta}{10}}\right) Qd^{\gamma} \quad (5)$$

Eqn 5 reveals a crucial aspect of sensing—namely, *scanning time increases super-linearly with increase in detection distance and linearly with span*. As described in Section 5.2, SpecNet uses this dependence to efficiently share load among spectrum analyzers given a scanning task.

To account for the deviations in scanning time from Equation 4 as depicted in Figure 5, given a detection range d , instead of using Eqn 5, SpecNet uses the lookup table $\tau(\rho)$ to determine the resolution bandwidth that has the fastest scanning time per MHz while ensuring a minimum noise floor of $P_0 - 10\gamma \log(d) - \Delta$. $P_0 = -50$ and $\Delta = 10$ are chosen as default unless specified by the user and $\gamma = 3$ is chosen as a conservative estimate.

Evaluation: Given a detection range, SpecNet chooses a resolution bandwidth so as to minimize scanning time. How well does the resolution bandwidth selection scheme work in practical deployments? A resolution bandwidth chosen too low will take too long to scan while a resolution bandwidth chosen too high will not provide the necessary SNR to allow detection. There are several practical considerations. First, the path loss exponent is not a fixed quantity and depends on the nature of the environment. Line of sight and non line of sight paths offer different path loss characteristics. Further, significant signal attenuation often occurs due to walls in indoor environments.

To answer this question, we tested SpecNet in a real deployment at the Indian Institute of Science (IISC) campus as depicted in Figure 7 on two different models of spectrum analyzer. The campus is lush with very dense trees and this provided an excellent opportunity to evaluate SpecNet in various scenarios such as Line of Sight (LOS), Non-Line of Sight (N-LOS) and Indoors. In Figure 7, two different models of spectrum analyzer are located at O, while a wireless microphone was placed at six different locations, two each in the LOS, NLOS and indoor categories. In each of the six detection experiments, the detection range was set to the exact distance between the microphone and the spectrum analyzer. P_0 was set to -35 dBm which was determined by measuring the power of microphone at a distance of 1m. For all our experiments we fixed $\Delta = 10$ dB. In other words, given a detection range, SpecNet must choose the resolution bandwidth that provides the minimum scanning time while ensuring that the SNR is a minimum of 10 dB. Table 8 provides a summary of the results.

Line of Sight: As seen from Table 8, for both the LOS experiments and for both spectrum analyzers, SpecNet chose a very conservative noise floor—while the target SNR is 10 dB, the observed SNR is about 25 dB. Figure 6 depicts the decay of signal strength with distance for the microphone in line of sight. The path loss decay exponent γ was estimated to be around 2.5, however, SpecNet conservatively chooses $\gamma = 3.0$ in estimating the target noise floor. This results in the conservative choice of the resolution bandwidth.

Non Line of Sight: For *NLOS experiments*, the resolution bandwidth choice of SpecNet allows for an SNR close to the target 10dB for both spectrum analyzers indicating that γ was closer to 3 for these experiments.

Indoors: When the microphone was kept *indoors*, however, SpecNet finds itself underestimating the signal decay. For example, in both the experiments, the chosen resolution bandwidths allow only SNR of about 6 dB rather than 10 dB.

While choosing a conservative resolution bandwidth ensures detection, it results in longer scanning times. What is the loss in scanning time due to the conservative choices of resolution bandwidth? To answer this question, we attempted to detect the microphone at sev-

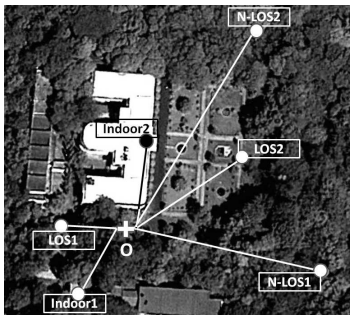


Figure 7: Occupancy detection using a single spectrum analyzer

	Distance in mts	SNR in dB		Loss in Sec Scanning Time	
Line Of Sight	31	24	28	0.005	0.018
	71	25	28	0.016	0.046
Non-Line Of Sight	124	15	23	0.123	2.23
	131	17	24	0.123	2.24
Indoor Locations	35	16	6	0.005	0.0
	50	0.2	8	0.0	0.0

Figure 8: Performance of Resolution Bandwidth Selection in SpecNet; the two columns for SNR and Scanning time represent two different spectrum analyzers

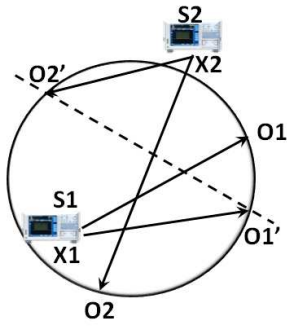


Figure 9: Occupancy detection using two spectrum analyzers

eral different resolution bandwidths without the use of SpecNet's resolution bandwidth selection. We then determined the optimal resolution bandwidth for each experiment that allowed an SNR of 10 dB. Table in figure 8 depicts the loss in scanning time in seconds due to the sometimes conservative choice of SpecNet for each experiment. As seen from table, the loss in scanning time is in the range of a few milliseconds most of the time and up to a few seconds in some cases. Thus, we conclude that the automatic resolution bandwidth estimation in SpecNet works as intended.

5.2 Occupancy Detection

In many practical applications of occupancy detection, users are interested in spectrum occupancy in a specific geographic region. For example, "are there any ongoing transmissions in the spectrum range 700 MHz to 800 MHz within a 5 km radius of my location?" SpecNet allows users to specify a circular region specified by a center and a radius for spectrum measurement. Behind the scenes, SpecNet determines the set of relevant spectrum analyzers that can be used to accomplish this task. Any spectrum analyzer whose maximum detection range (determined by the lowest resolution bandwidth) overlaps with the user-specified region of interest is deemed relevant. When there are multiple relevant spectrum analyzers, SpecNet schedules the scanning task load among them so as to minimize the overall scanning time.

5.2.1 Load sharing across multiple spectrum analyzers

There are two distinct dimensions along which a scanning task can be shared among multiple spectrum analyzers, namely, spectrum and geography. *Spectral load sharing* involves different spectrum analyzers scanning complementary parts of the spectrum while *geographical load sharing* involves different spectrum analyzers scanning different spatial sections of the overall geographical area of interest. SpecNet uses a combination of both these techniques to minimize overall scanning time.

The Scheduling Metric: If n different spectrum analyzers are scheduled to share a certain task load, they scan in parallel and accomplish their respective sub-tasks in parallel. Suppose that the i^{th} spectrum analyzer takes time T_i to complete its assigned sub-task. The task is deemed complete when all spectrum analyzers have accomplished their respective sub-tasks. Since all spectrum analyzers are tasked in parallel, the time to task completion is given by $T = \max(T_1, T_2, \cdots, T_n)$. The goal of the SpecNet task scheduler is to minimize the task completion time. Hence, SpecNet attempts to schedule various spectrum analyzers in such a manner that the maximum over all sub-task completion tasks is minimized *i.e.,* in a *min-max* manner.

Spectral Load Sharing: Figure 9 depicts a circular region of interest and two spectrum analyzers S1 and S2 located at X1 and X2 that can potentially be used to scan the circular region of interest. Suppose that the user needs to scan from f_{min} MHz to f_{max} MHz. S1 and S2 could then share the task such that S1 scans from f_{min} MHz to $f_{min}+Q_1$ MHz, while S2 scans from $f_{min}+Q_1$ MHz to f_{max}. Such spectral load sharing results in a reduction in span for the participant spectrum analyzers, thus reducing the overall scanning time.

In the above example Q_1 must be chosen in a manner so that the maximum of the scanning times of S1 and S2 are minimized. In order to detect any transmission in the entire region of interest, S1 must have a detection range equal to $|\overline{X_1O_1}| = d_1$ where O1 corresponds to the farthest possible transmitter location within the region of interest from S1 (as depicted in Figure 9). Similarly, the detection range of S2 should be $|\overline{X_2O_2}| = d_2$ in order to detect any transmitter in the region of interest. Let τ_i be the minimum scanning time per MHz for spectrum analyzer S_i required to achieve a detection range of d_i. Then the overall scanning time is given by $\max(\tau_1 Q_1, \tau_2 Q_2)$, where $Q_2 = f_{max} - f_{min} - Q_1$. The optimal choice then corresponds to when

$$Q_1 : Q_2 = \frac{1}{\tau_1} : \frac{1}{\tau_2} \tag{6}$$

Eqn 6 can be easily generalized to spectral partitioning for several spectrum analyzers. In case of several spectrum analyzers, *the span of spectrum allocated to each spectrum analyzer is inversely proportional to the minimum scanning time per MHz required to scan the circular region of interest.*

Geographical Load Sharing: Another way to share the load between S1 and S2 (Figure 9) is to partition the region of interest geographically by requiring them to scan only parts of the region of interest rather than the entire region. In Figure 9, the region is divided into two sections by the line $|\overline{O_1'O_2'}|$. S1 and S2 are deemed responsible to scan each of the two sections. The advantage of partitioning in this manner is that individual spectrum analyzers can now use a smaller detection range. As seen in Figure 9, S1 and S2 use detection ranges equal to $|\overline{X_1O_1'}| = d_1' < d_1$ and $|\overline{X_2O_2'}| = d_2' < d_2$ respectively. As described in Equation 5, reduced detection range implies reduced scanning time. Thus, each of the spectrum analyzers takes a shorter time to scan its respective region—thus reducing overall task completion time.

Since every spectrum analyzer scans a different geographical region, each must scan the entire spectrum of interest f_{min} to f_{max}. If the scanning times per MHz of n geographically task sharing spectrum analyzers are given by τ_1,τ_2,$\cdots$,τ_n, then the over all task completion time will be $\max(Q\tau_1, Q\tau_2, \cdots, Q\tau_n)$. Consequently, in order to minimize over all task completion time, we need $\tau_i = \tau$, $\forall i$ such that τ is minimized while ensuring that the entire area of interest is covered.

First consider the case of homogeneous spectrum analyzers. Ensuring equal τ_i translates to ensuring equal maximum detection ranges to all the spectrum analyzers. This problem can be optimally solved using Voronoi partitioning with each spectrum analyzer being treated as a Voronoi site. Each Voronoi cell, then, would correspond to the geographical region assigned to the spectrum analyzer. The resolution bandwidth of each spectrum analyzer would correspond to the detection range required to accommodate the farthest point in its Voronoi cell.

Now consider the case of heterogeneous spectrum analyzers. Since the scanning times of different analzyers are different, standard Voronoi partitioning is no longer optimal. Instead, the SpecNet scheduler performs a modified version of Voronoi partitioning – equal detection time partitioning – where proximity is measured in terms of detection time rather than Euclidean distance.

Given the non-linear and discontinuous nature of dependence of detection time on detection range (Equation 5), to the best of our knowledge there exists no known exact solution to this partitioning problem. Consequently we resort to solving the problem numerically. The entire area of interest is sampled at several locations generated randomly over the area of interest. Each random location is then assigned to its nearest spectrum analyzer in terms of the scan-time required to detect a transmitter at that grid point. Note that if a point is located beyond the detection range of a spectrum analyzer, the corresponding scanning time is set to infinity. Finally, each spectrum analyzer is assigned a resolution bandwidth by setting its detection range to the farthest random location assigned to it. The run-time complexity of this numerical scheme depends on the number of random points chosen. In our implementation we generated random locations with a density of 1 location per sq meter. For an area of 1 Sq Km (1×10^6 random locations) we found that geographic partitioning took under a few hundred milliseconds on the SpecNet server.

5.2.2 Geographical versus Spectral Load Sharing

Which of the above two load-sharing schemes should we use and under what circumstances? To answer this question we describe the results of two experiments conducted in the Indian Institute of Science (IISC) campus, depicted in Figures 10a and 10b, scanning from 700-800 MHz. In each of the experiments we compared three different scheduling methods. In *Best Select*, the spectrum analyzer that can accomplish the task in the shortest time is selected and used to accomplish the scanning task without any load sharing. We compared Best Select with spectral and geographical load sharing.

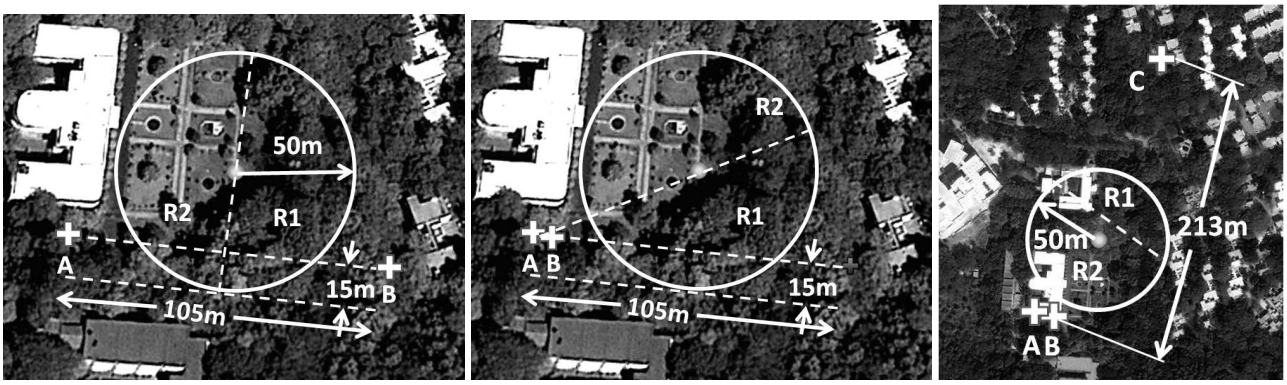
Experiment I : Two identical spectrum analyzers (both N9320B Agilent models) were placed 103 m apart at A and B as depicted in Figure 10a. The region of interest was specified as a circle of radius 50 m.

Experiment II : Two identical spectrum analyzers (both N9320B Agilent models) were both placed at location A and the region of interest was specified as a circle of radius 50 m as shown in Figure 10b.

Experiment	Best Select in sec	Spectral in sec	Geographical in sec
Experiment I	1054	561	129
Experiment II	1054	561	1054

Table 2: Comparison of load sharing schemes

Results of Experiment I : As depicted in Table 2, since the spectrum analyzers are identical, the optimal spectral load sharing resulted in both the spectrum analyzers taking an almost equal amount of time (in practice a slight difference in their noise floors resulted in one spectrum analyzer scanning a bit more spectrum than the other). Consequently, spectral partitioning completed about twice as fast as Best Select. Curiously, *geographical load sharing completed almost five times faster than spectral load sharing*. In this particular experiment, Voronoi partitioning resulted in two halves of the circle indicated by regions R1 and R2 in Figure 10a. Conse-



(a) Experiment I (b) Experiment II (c) Experiment III

Figure 10: Comparison of scheduling schemes

quently, the detection range required for each of the spectrum analyzers in geographical load sharing was smaller than that required in spectral load sharing. Eqn 5 reveals that scanning time decreases super-linearly as detection range, explaining the 5x gains.

Results from Experiment II : As depicted in Table 2, since the spectrum analyzers are co-located and identical, optimal spectral load sharing assigns two halves of the span to each spectrum analyzer. Consequently, spectral load sharing performs approximately twice as well as scheduling without load sharing. Here, however, geographical load sharing performs exactly the same as having no load sharing and takes twice as long as spectral partitioning! The Voronoi partition for the experiment is indicated by the dashed line separating R1 and R2 in Figure 10b. The maximum detection range required by each of the two spectrum analyzers to cover their respective partitions is actually almost the same as that required to cover the entire circular region of interest. Since both the spectrum analyzers scan the entire spectrum, one of the spectrum analyzers is actually redundant. *This experiment shows that when spectrum analyzers are very closely located, spectral partitioning can be more advantageous than geographical partitioning.*

5.2.3 Geo-Spectral Load Sharing

Spectral and geographical task sharing, as described in Section 5.2.1, each optimize along a single dimension only, namely either frequency (spectral) or area (geographical). As seen from Experiments I and II (Section 5.2.2), while geographical task sharing may be superior to spectral in some scenarios, the opposite may be true in others. A more general task partitioning scheme then is *geo-spectral* partitioning—where optimization is performed simultaneously along both the spectral and geographic dimensions.

Optimal geo-spectral task sharing, where spectrum analyzers are assigned a combination of frequency range and geographical area to minimize overall task completion time while ensuring that the entire area and spectrum of interest are covered, falls under a class of non-convex optimization problems for which, to the best of our knowledge, there exists no known exact solution. However, Experiments I and II (Section 5.2.2) reveal two key observations that allow us to develop a heuristic to enable geo-spectral task sharing. First, geographical partitioning typically out-performs spectral partitioning owing to the super-linear relationship between detection range and scanning time. Second, when spectrum analyzers are located near each other, spectral partitioning tends to outperform geographical partitioning.

In order to facilitate explanation of our heuristic for geo-spectral task sharing, we introduce the notion of a spectrally sharing cluster (SSC) of spectrum analyzers – a set of spectrum analyzers that share their scanning tasks spectrally over the same geographical region (possibly over only a small part of the entire region of interest). An SSC can be replaced by a single representative Virtual Spectrum Analyzer (VSA). The distance of a location from this VSA is then the maximum over the distances all spectrum analyzers in the corresponding SSC, since even the farthest constituent spectrum analyzer must detect occupancy at this location. The occupancy detection time for any location using the VSA is determined by optimally partitioning the spectrum among the constituent spectrum analyzers in the corresponding SSC (as described in Section 5.2.1). The union of two SSCs yields a VSA comprising the union of all constituent spectrum analyzers in both SSCs.

Our geo-spectral task sharing heuristic for n spectrum analyzers is initialized by creating n SSCs, each comprising a single distinct spectrum analyzer and performing geographical task sharing on them. The algorithm is a greedy iterative scheme, where at each step, pairwise SSC unions are considered in order to determine if overall task completion time can be reduced. In order

Model	Frequency Range	RBW steps
Agilent N9320B	9 KHz- 3 GHz	11 (10 Hz - 1 MHz)
Agilent Fieldfox N9912A	5 KHz - 6 GHz	36 (10 Hz - 1 MHz)
Agilent EXA N9010A	9 KHz - 26.5 GHz	62 (1 Hz - 8 MHz)
Agilent PSA E4440A	3 Hz - 26.5 GHz	68 (1 Hz - 8 MHz)
Hewlett-Packard E4403B	9 KHz- 3 GHz	15 (10 Hz - 5 MHz)

Table 3: Spectrum analyzer models used in SpecNet

to determine overall task completion time given a set of SSCs, each SSC is replaced by its corresponding VSA and geographical partitioning is performed on this set of VSAs. The SSC pair union that results in the maximum reduction in overall task completion time is accepted for the next iterative step. The procedure continues until no further opportunities to unite SSCs exist that can reduce the overall task completion time. In the worst case, the algorithm terminates in n steps, as at each step the number of SSCs decreases by 1. As, at each step all pairs of SSCs must be explored, the worst-case running time of this algorithm is $O(n^3)$. Since spectral sharing typically yields benefits only when two spectrum analyzers are "close", in practice the running time can be reduced to $O(n^2)$ by considering a fixed number of closest SSCs rather than all possible SSC pairs at each step.

Figure 10c depicts an example of Geo-Spectral load sharing. The scanning frequency range was chosen as 700 MHz to 800 MHz. Spectrum analyzers S1, S2 and S3 are located at A, B and C respectively. S3 (Fieldfox) is a much faster spectrum analyzer compared to S1 and S2 (both N9320B Agilent). The circular region of interest is geographically partitioned into two regions R1 and R2. S1 and S2 scan region R1 using spectral load sharing while S3 scans the entire spectrum in geographic region R2. To compare the performance of geo-spectral partitioning we also tried scheduling using the purely geographic and spectral schemes. Geographic load sharing took 1205 seconds; spectral load sharing 1118 seconds; and geo-spectral load sharing only 526 seconds.

In summary, load sharing across multiple spectrum analyzers is a challenging problem. SpecNet's Geo-Spectral load sharing algorithm is able to achieve 2-5X speedup compared to using a single spectrum analyzer in our experiments.

6 Implementation

The SpecNet platform is accessible at [15] via a web service API. It consists of a *master server* that manages several *slave servers*.

6.1 Master Server

The *master server* performs two major functions—first, it exposes an API (Section 4) which the SpecNet clients/users utilize to write programs and second, it manages all the *slave servers* connected to it.

As mentioned in Section 4, the API is exposed as XML-RPC calls to allow access from a wide-range of platforms. The *master server* implements a push-based model and thus, TCP connections to the slave servers are kept persistent using heartbeats. The current implementation of the master server is centralized and consists of approximately 5000 lines of C# code. However, partitioning of the slave servers along geographic boundaries is possible, thus allowing distributed execution across multiple master servers if scalability concerns arise.

One of the key challenges in managing slave servers is dealing with the heterogeneity of spectrum analyzers. As shown in Table 3, spectrum analyzers differ in their supported resolution bandwidth steps and frequency range of operation. Further, as discussed earlier, scan times (Figure 5) and noise floor (Figure 2) also vary across spectrum analyzers. SpecNet accounts for each of the above variations through a novel, automatic remote benchmarking process, described in detail in [8], that allows the master server to quickly build up a lookup table of scan times and noise floor values at different resolution bandwidth steps for each of its slave servers.

6.2 Slave Servers

The *slave server* is a small piece of software that runs on a desktop or laptop that are directly connected to the spectrum analyzer. The main task of the *slave server* is to act as a bridge between the spectrum analyzer connected to it and the *master server*. To avoid issues with NAT/-firewalls, the *slave server* initiates an outbound TCP connection on port 22 to the *master server*. It also connects to the local spectrum analyzer through VISA. Once connected, it translates commands from the master server to the spectrum-analyzer-specific-commands, runs spectrum scans, and returns the results.

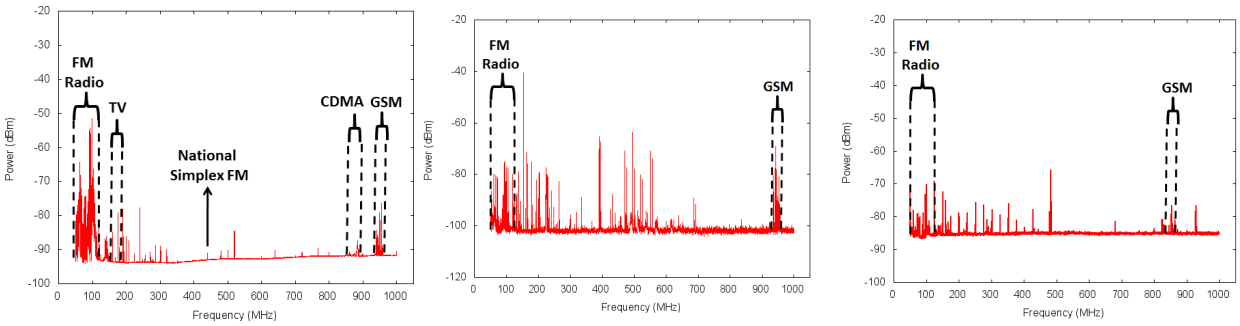
In order to support multiple platforms, we have implemented the slave server in Python in approximately 1000 lines of code. We use the PyInstaller [13] package to generate platform specific (Windows & Linux as of today) executables.

7 Applications

In this section, we present three example user applications on the SpecNet platform that highlight the simplicity of building a networked, geo-distributed system of spectrum analyzers.

7.1 Remote Spectrum Measurement

In this section we demonstrate how SpecNet can be used to make spectrum measurements anywhere in the world. The user code fragment written in Python is shown in Listing 1. One simply needs to connect to the SpecNet server, identify available devices in the region of interest and then use the `GetPowerSpectrum()` API to obtain power values in the desired parts of the spectrum. This data can be used, for example, to compare available free spectrum in different parts of the world or as



(a) Bangalore, India (b) Edinburgh, UK (c) Stony Brook, USA

Figure 11: Spectrum occupancy in various geographic regions

traces for evaluation of new white-space protocols such as WhiteFI [3].

Listing 1: Code snippet for remote measurement.

```
# connect to SpecNet server
apiServer = xmlrpclib.ServerProxy(
  "http://bit.ly/SpecNetAPI",
  allow_none=True);

# Find devices from region of interest
devices = APIServer.GetDevices(
  [55.944350, -3.187745, 500.0], None);
for device in devices:
    power_vals = APIServer.GetPowerSpectrum(
     device['ID'], Fs, Fe, 1e3);
```

At the time of writing, in addition to a few spectrum analyzers in Bangalore (India), we had one spectrum analyzer in Stony Brook (USA) and one in Edinburgh (UK) that were connected to SpecNet. Figure 11 shows the spectrum measurements at these three sites located in three different continents, demonstrating the world-wide reach of the SpecNet platform. As seen from Figure 11, spectrum measurements at each of these locations across the world clearly identify the well-known transmitters such as FM, TV, *etc.*, and the available spectrum whitespaces.

7.2 Primary Coverage

The next example application determines the spatial footprint of a TV transmitter located within a large city. This may be useful for whitespace network operators in planning their deployments. Determining the footprint of a TV transmitter invariably requires knowledge of its location. While accurate databases of these locations are available in countries such as the US, such a database is not readily available in many developing countries, including India. We tried to obtain this information by contacting the Indian government agencies via postal mail (under the Right-to-Information Act). While we received information on about 150 TV tower locations (out of an estimated 700 towers), we found many inaccuracies in the data. For example, one tower's location was mapped well into a bay! Upon analyzing this TV tower data for five cities (ground truth based on Wikimapia), we found localization errors to range between 2-83 km (average 22 km, median 5 km). We now highlight how SpecNet could be used as a low-cost solution to improve the coverage and accuracy of the existing TV tower database.

Listing 2: Code snippet for primary coverage.

```
# Get Spectrum Analyzers in region
area_of_interest = [13.02236,77.56558, 100000.0];
devices = APIServer.GetDevices(area_of_interest, None);

# Get Power Spectrum Values
for device in devices:
    power_vals = APIServer.GetPowerSpectrum(
        device['ID'], Fs, Fe, 1e3);
    power_vals.append(average(power_vals));
    observation_locations.append([device['latitude'],
          device['longitude']]);
# Localize
if number_of_locations < 5
    localization_res = APIServer.LocalizeTransmitter(
     area_of_interest, observation_locations,
     power_values, 'LDPL', [-35.0, 3.0]);
else
    localization_res = APIServer.LocalizeTransmitter(
     area_of_interest, observation_locations,
     power_values, 'LDPL', None);

# Interpolate
pow = APIServer.FindPowerAtLocation(new_location,
 [localization_res], 'LDPL', None);
```

The code snippet for this application is shown in Listing 2. The region of interest is identified and power spectrum values from devices in that region are obtained. Then the TV transmitter is localized using the `LocalizeTransmitter()` API. Finally, a path loss model is used to build the spatial footprint of the TV transmitter. The API `FindPowerAtLocation()` is then used to determine the received power at desired new locations.

Bangalore city has one terrestrial TV transmitter. For the purpose of evaluation in a large-scale setting, we needed data from multiple spectrum analyzers at different locations in the city. Also, the accuracy of the localization API depends on the number of measurement locations. However, at the time of evaluation we only had access to four slave servers inside Bangalore. To get

around this problem, we modified the master server to allow mobile slave servers to connect to it. This enabled us to gather data from multiple locations in the city using just one mobile slave server by driving on the major roads and highways of the city. Figure 12 depicts the locations in the city where measurements were collected.

Figure 13 shows the TV tower localization error mean, 25th and 75th percentile (y-axis) as the number of measurement locations are varied (x-axis). To generate each point in Figure 13, twenty subsets of locations were randomly picked from the set of all measurement locations. We see that even when the number of measurement locations is between 5-10, the mean localization error varies between 2.5-3.8 km. This demonstrates that even by using measurements from a small number of spectrum analyzers in each city, the gaps and inaccuracies in the government database can be corrected significantly.[1] As the number of measurement locations is increased to 100, we see that the localization error goes below 0.5 km. While it is unrealistic to assume that SpecNet would have over 100 spectrum analyzers in each city, an alternative is to have spectrum analzyers that are mobile as part of SpecNet—we plan to look into this in the future.

Figure 14 shows the mean, 25th and 75th percentile errors in signal strength predictions obtained by using the interpolation API. The mean signal error varies between 6 to 8 dB, similar in magnitude to the expected signal variations due to the environment.[2] Thus, using SpecNet to calculate coverage of a primary transmitter can provide a good estimate to an operator.

7.3 SpectrumCop

Our final application demonstrates the two key features of SpecNet: 1) simplicity of writing a complex real-time application through the use of high-level APIs and 2) efficiency of SpecNet in scanning a wide frequency range when more than one spectrum analyzer is available, in order to detect violators quickly.

The goal of this application is to quickly detect a static narrow-band transmitter within a certain geographical region of interest and then localize the transmitter. The transmitter can be operating anywhere within a wide frequency range. This application is especially useful for, say, government officials to monitor unauthorized transmitters in a certain band.

The code snippet for this application is shown in Listing 3. The application uses the `GetOccupancy()` API for the transmitter detection part, which basically tasks one or more spectrum analyzers in the vicinity to perform scans at an appropriate resolution bandwidth and frequency range. The result of this API call is an occupancy list, which indicates frequencies that have ongoing transmissions. A more detailed spectrum measurement is then performed only in the region around the detected frequency. The results of the scan are then fed to the `LocalizeTransmitter()` API to determine the location of the transmitter.

Listing 3: Code snippet for SpectrumCop.

```
# Find occupancy in desired region
bound = [lat, lng, radius];
options = [lat, lng, radius, min_power_to_detect];
occupancy_list = APIServer.GetOccupancy(bound,
 start_frequency, end_frequency, min_power_detect);

# Get power spectrum for transmitter frequency
for occupancy in occupancy_list:
    if (occupancy['Occupied'] == 1):
        new_f_start = occupancy['Frequency'] - 250e3;
        new_f_end = occupancy['Frequency'] + 250e3;
        devices = APIServer.GetDevices(bound, None);
        for device in devices:
            locs.append([device['Latitude'],
                         device['Longitude']]);
            results[device['ID']] = APIServer.
             GetPowerSpectrum(device['ID'],
             new_f_start, new_f_end,
             options); # Actual call in new thread.
        break;

# Localize transmitter based on power measurements
for r in results:
        powers.append(max(r));
print APIServer.LocalizeTransmitter(bounds, locs,
 powers, 'LDPL', [P, 3.0]);
```

Evaluation: We used this application to detect and localize a microphone in a region of 75 meters radius in IISc. The setup consisted of 3 spectrum analyzers that were placed near 3 corners of the region of interest. The microphone transmits in a 250 KHz narrow band and the frequency range of the search space is set to 3 MHz. The SpectrumCop application detected the microphone perfectly and localized it to within 20 meters of the actual location. The entire process of detecting and locating the microphone took 165 seconds.

8 Limitations

First, spectrum analyzers are expensive equipment that researchers have procured for specific needs. It may not be easy to convince owners to volunteer this resource to the community, especially during the bootstrapping stage where the benefit of the platform is not clear to the owner. To date, we have approached a few of our acquaintances and have observed mixed results. In the long run, perhaps governments may be willing to sponsor a set of spectrum analyzers dedicated for SpecNet use.

Second, spectrum analyzers are typically used inside labs that may be in basements or deep inside buildings. Our measurements indicate that buildings can add 5-20 dB of attenuation (20dB in the basement for FM/TV

[1] Note that we used basic triangulation to locate the T.V tower, it may be possible to achieve a higher accuracy through more sophisticated localization schemes proposed in literature.

[2] In our implementation we used a simple log distance path loss model. The use of more sophisticated path loss models such as those that use terrain information may provide more accurate predictions

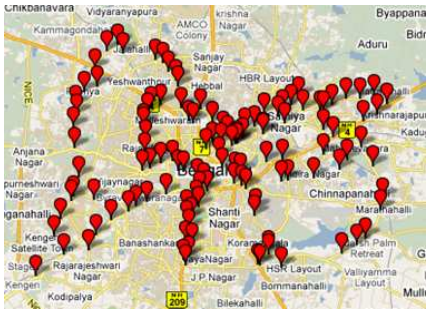


Figure 12: Measurement locations

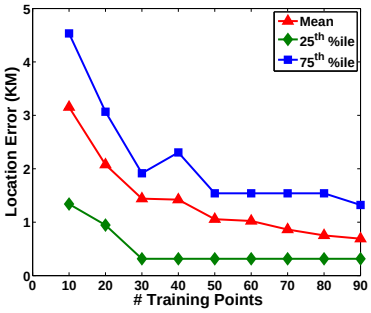


Figure 13: TV Tower Localization

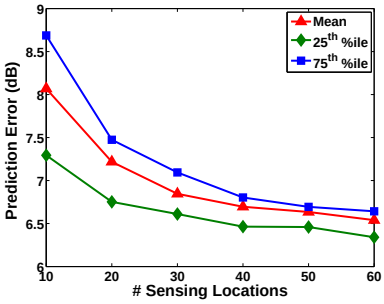


Figure 14: Interpolation results

transmissions) which restricts the detection range of the analyzer. If the owner can be convinced to mount the antenna near a window, the utility of the spectrum analyzer can be significantly increased. To minimize variability due to antenna placements, SpecNet can choose to only include spectrum analyzers with unobstructed antennas.

Finally, we have not considered the privacy/security implications of allowing remote scanning of the spectrum. For now, SpecNet only exposes the power values measured from the spectrum scan. Thus, it prevents direct security and privacy threats such as fine-grained traffic monitoring or user tracking. Advanced spectrum analyzers can provide time domain (I/Q) samples of the scan and support for these features in SpecNet would require sophisticated controls for privacy and security.

9 Conclusion

After the FCC ruling in the U.S. allowing opportunistic access to portions of licensed frequency bands, there has been tremendous interest in both academia and industry in developing novel wireless techniques and products that take advantage of the new rules. A key requirement for enabling this new ecosystem is a measurement infrastructure that can provide real data. SpecNet fulfills this critical need by enabling geographically distributed spectrum analyzers to be networked, thereby allowing both real-time remote measurements as well as collection of historic spectrum usage data. Furthermore, SpecNet exposes an API that allows users to build interesting distributed sensing applications like SpectrumCop with relative ease. There is still a lot of work left to achieve our goal of building a planet-scale networked spectrum analyzer testbed, but we believe SpecNet provides a good base to build upon.

10 Acknowledgements

We thank our shepherd, Brad Karp, and the anonymous reviewers for their constructive comments. We also thank Arsham Farshad, Mahesh Marina, and Akshay Athalye for helping us conduct remote spectrum measurements.

References

[1] http://www.emulab.net/.
[2] AMMARI, H., AND DAS, S. Promoting heterogeneity, mobility, and energy-aware voronoi diagram in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems 19*, 7 (jul. 2008), 995 –1008.
[3] BAHL, P., CHANDRA, R., MOSCIBRODA, T., MURTY, R., AND WELSH, M. White space networking with wi-fi like connectivity. *ACM SIGCOMM* (2009).
[4] CHEN, D., YIN, S., ZHANG, Q., LIU, M., AND LI, S. Mining spectrum usage data: a large-scale spectrum measurement study. In *ACM MobiCom* (2009).
[5] CHIANG, R., ROWE, G., AND SOWERBY, K. A quantitative analysis of spectral occupancy measurements for cognitive radio. In *Vehicular Technology Conference* (2007).
[6] GANESAN, G., AND LI, Y. Cooperative spectrum sensing in cognitive radio networks. In *DySpan* (2005), IEEE.
[7] ISLAM, M., KOH, C., OH, S., QING, X., LAI, Y., WANG, C., LIANG, Y.-C., TOH, B., CHIN, F., TAN, G., AND TOH, W. Spectrum survey in singapore: Occupancy measurements and analyses. In *CrownCom* (2008).
[8] IYER, A., CHINTALAPUDI, K., NAVDA, V., RAMJEE, R., PADMANABHAN, V., AND MURTHY, C. Specnet: Spectrum sensing sans frontieres. Tech. rep., Microsoft Research, Feb 2011.
[9] MCHENRY, M. A. NSF Spectrum Occupancy Measurement Project Summary. In *Shared Spectrum Company Report* (2005).
[10] MCHENRY, M. A., TENHULA, P. A., MCCLOSKEY, D., ROBERSON, D. A., AND HOOD, C. S. Chicago spectrum occupancy measurements & analysis and a long-term studies proposal. In *TAPAS* (2006).
[11] MISHRA, S. M., SAHAI, A., AND BRODERSEN, R. W. Cooperative Sensing Among Cognitive Radios. In *ICC* (2006), pp. 1658–1663.
[12] http://www.planet-lab.org/.
[13] PYINSTALLER. http://www.pyinstaller.org/.
[14] SCPI. http://www.ivifoundation.org/docs/SCPI-99.PDF.
[15] SPECNET WEBSITE. http://bit.ly/SpecNet.
[16] UNNIKRISHNAN, J., AND VEERAVALLI, V. Cooperative spectrum sensing and detection for cognitive radio. In *GLOBECOM* (2007), IEEE.
[17] VIERA, M., VIERA, L., RUIZ, L., LOUREIRO, A., FERNANDES, A., AND NOGUEIRA, J. Scheduling nodes in wireless sensor networks: a voronoi approach. In *LCN* (2003), IEEE.
[18] WERNER-ALLEN, G., SWIESKOWSKI, P., AND WELSH, M. MoteLab: A Wireless Sensor Network Testbed. In *IPSN* (2005).
[19] WILLCOMM, D., MACHIRAJU, S., BOLOT, J., AND WOLISZ, A. Primary Users In Celular Networks : A Large Scale Measurement Study. In *DySPAN* (Oct 2008).
[20] YANG, L., HOU, W., CAO, L., ZHAO, B. Y., AND ZHENG, H. Supporting demanding wireless applications with frequency-agile radios. In *NSDI* (2010), USENIX.