

A more general whitespace architecture: refactoring the master-client paradigm

Kate Harrison and Anant Sahai

Wireless Foundations, EECS, UC Berkeley
{harriska, sahai}@eecs.berkeley.edu

Abstract—While some whitespace devices will be self-sufficient (“masters”), others will rely on help from other devices in order to access the whitespaces (“slaves”). Currently, this help is provided by a single master device. In this paper, we argue that (1) this assistance need not be provided by a single device and (2) the assisting device need not be a whitespace device. Instead, we can think of the “slave” as being helped by a whitespace device support network, i.e. a variety of devices which each supply a piece of the whitespace access puzzle.

We begin by identifying the three key components of a whitespace device support network. We describe each component in detail before giving example deployments which are only possible with a support network. In one example, a smartphone plays the role of the “master” by providing the “slave” device with a location as well as a means to access the database.

Finally, we remark on the advantages that this separation provides when it comes to certification. In particular, regulators can now perform unit tests to verify that each component operates correctly on its own, rather than certifying an entire device all at once.

I. INTRODUCTION

The US’s Federal Communications Commission (FCC) significantly advanced spectrum regulation by opening up what are known as the TV whitespaces in 2008 [1]. These unused pieces of spectrum are numerous and potentially of high social and economic value [2]. Other regulators, such as Ofcom in the UK, are also working toward similar regulations [3].

The United States’ President’s Council of Advisors on Science and Technology (PCAST) argued in 2012 that dynamic spectrum access is essential, stating that “the traditional practice of clearing government-held spectrum of Federal users and auctioning it for commercial use is not sustainable” and recommending that the Secretary of Commerce “identify 1,000 MHz of Federal spectrum in which to implement shared-use spectrum pilot projects” [4].

In this paper, we refer to all spectrum-sharing devices that defer to primary users as whitespace devices (WSDs). Regardless of the band or their priority in it, these WSDs need a mechanism to ensure that they do not cause interference.

However, it is not enough to simply open up spectrum for sharing: it must also be usable. Unnecessarily restrictive regulations (current and future) may inadvertently affect the design of the whitespace devices, thus artificially limiting their potential and blocking innovation. To that end, regulatory agencies have attempted to build a high degree of flexibility into their regulations.

Even as early as 2008 [1], the FCC recognized databases as an important method for discovering whitespaces¹. It has since been shown that databases are actually the only certifiable method for discovering whitespaces [5], and the research efforts of the spectrum sharing community generally reflect that fact. However, the database method also requires some form of geolocation capability in order to accurately query the database. This means that devices lacking geolocation capability—either simple/cheap devices or GPS-reliant devices operating indoors—are left without any acceptable options.

Recognizing this limitation, the FCC (and Ofcom) introduced two modes of operation for devices: Mode I (client a.k.a. slave) and Mode II (master), saying “we believe that this approach will provide flexibility to permit a wide range of unlicensed broadband uses and applications” [1, §54].

This separation of responsibilities—geolocation and database access from actual operation—represents a significant step toward a more general architecture. The flexibility inherent in the existing regulations allows for a variety of devices that would have been out of the question with the naive approach; for example: low-power wireless sensor networks, *all* indoor devices, and widely-dispersed decentralized wireless control systems.

However, we argue in the rest of this paper that the separation of responsibilities is not yet complete. In particular, we recommend that the essential components of whitespace systems be recognized as *individual* components which can be combined in a variety of ways. We do not propose to add new components to the architecture, merely to call out and separate the components which already exist in today’s regulations. These components are:

- 1) The ability to determine the device’s location
- 2) The ability to communicate with the database
- 3) The software-defined ability to assemble a database request and understand the response

The critical observation is this: *these components need not be part of the same device, nor even part of a whitespace device.*

¹Using the database method, a secondary device intending to operate in the whitespaces must communicate its (approximate) location to a whitespace database (WSDB). The WSDB then calculates, using data and regulations provided by the regulator, the operating parameters which will be safe (from the incumbent’s perspective) for the secondary device to use. These parameters, which include frequencies on which the device may transmit, are then communicated to the secondary, after which it may commence operation.

We see several advantages to this proposal:

- Certification of individual components (and, separately, their interactions) is much simpler and more trustworthy than certifying an entire, complex device.
- Whitespace devices from all bands will have the opportunity to share a common support infrastructure in order to amortize costs and enhance scalability. This is consistent with the PCAST vision of widespread spectrum sharing. In fact, our vision takes it a bit further: while PCAST envisions only a common database, we see a much richer common infrastructure emerging naturally [4].

The title of this paper refers to a common software engineering practice called “refactoring,” in which old code is rewritten to improve design without changing external behavior. We see our proposal as a “refactoring of regulations” which reduces complexity while increasing performance and confidence. This refactoring does not change the protection guarantees offered to primary users or the political balance struck among non-technical social objectives.

The organization of this paper is as follows. In Section II, we describe the key components of whitespace devices and in Section III we give examples which motivate their separation. In Section IV, we connect this separation to the well-established software engineering principles of modularity, testability, and scalability. In Section V, we discuss the fortuitous certification consequences of our proposed architecture. Finally, we close by discussing some compelling byproducts of our proposed architecture, such as the economic incentives.

II. ESSENTIAL COMPONENTS OF WHITESPACE SYSTEMS

A whitespace device (WSD) is a device that operates under particular time-space-frequency constraints designed to protect devices with higher priority. This definition is irrespective of the type of primary as well as the WSD’s priority in the band. It is with this broad definition in mind that we proceed to identify the universal goals and components of WSDs.

A. Universal goals of whitespace devices

Throughout this paper, we will assume that whitespace devices and their manufacturers have the following values:

- *Equipment should be (at least potentially) inexpensive.* In particular, regulations should avoid requiring that devices add components (cost) purely for the sake of regulatory compliance. If it becomes too costly to use the whitespaces, the ISM (i.e. internationally-available unlicensed) bands or even paid spectrum may be a better choice for many companies.
- *Whitespaces should be accessible/recoverable under reasonable conditions.* For example, devices should not be relegated to the ISM bands when indoors, nor should they by default lose out on many whitespace opportunities.

Our work strives to enable manufacturers to meet their goals in reasonable and flexible ways. We also aim to enable and promote application-agnostic regulations.

B. The whitespace access problem

The primary goal of whitespace regulations is to avoid interfering with primary users and the secondary goal is to make as much dormant spectrum as possible available to secondary users. What capabilities do whitespace devices (WSDs) need to achieve these goals? Devices need to:

- 1) Gather location-relevant information (“localization”)
- 2) Communicate location-relevant information to the whitespace database and to receive operating parameters in return (a “gateway”)
- 3) Bridge the above services, resulting in a simple set of operating parameters (“packager”)

Under the current master-client paradigm, a device either has all of these capabilities (master) or it has none of them (client). Specifically, the master is assumed to have geolocation capability and access to a whitespace database (WSDB).

We wish to consider the possibility that these responsibilities could be spread over multiple devices in a variety of configurations instead of using the restrictive master-or-slave dichotomy. We describe these components in detail in the following sections. For reference, Figure 1 lists the responsibilities and certification requirements for each component.

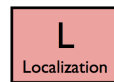
C Client	<ul style="list-style-type: none"> • Communicate device ID to packager ★ Wait to transmit until fresh operating parameters are received ★ Operate within the operating parameters
G Gateway	<ul style="list-style-type: none"> • Communicate with the packager (e.g. using Bluetooth or WiFi) • Communicate with the whitespace database (e.g. over IP)
L Localization	<ul style="list-style-type: none"> ★ Determine absolute location (if applicable) ★ Tell the time ★ Create and sign messages with the above info. • Transmit these messages (e.g. WiFi, whitespace broadcast)
P Packager	<ul style="list-style-type: none"> • Receive messages from a localization service ★ Create and sign a message containing localization information and client device ID • Communicate with the client and gateway

Key:

- Useful for the client
- ★ Certification required

Fig. 1. Responsibilities of each component

C. Localization



The localization service provides information that can be used to approximate a client’s location. We will elaborate on the idea of approximate location in Section III-E, but localization equipment could take the following forms:

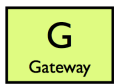
- A nearby WSD with its own geolocation capability.
- A transceiver on the roof of a building with a GPS unit installed. This device could serve indoor master-type users as well as nearby client-type users.
- A beacon which uses time-of-flight measurements to determine the device’s location relative to the beacon (used in conjunction with other localization equipment).

- A nearby smartphone running certified software
- A nearby GPS navigator running certified software.

The important thing is that the protocol be able to provide the database with a set of *possible* locations. The protocol need only guarantee that the client is within this “uncertainty region.” We detail how the database can easily compute safe operating parameters given any uncertainty region in [6].

Notice that there is nothing inherently whitespace-y about a localization service. Location is a function of space, not frequency. This means that localization services could serve devices in a variety of bands and not just for whitespace access. Thus, as dynamic spectrum access becomes prevalent and more localization services are deployed, even old devices will enjoy increased quality of service. So the amount of easily-recoverable whitespace increases with the need for it.

D. Gateway

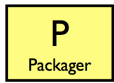


A gateway should provide a way of securely communicating with the whitespace database (WSDB). It only needs to act as a relay, without performing any computations of its own. In fact, communications with the WSDB should be encrypted to prevent the gateway from tampering with the information therein. To reiterate, a gateway should act as a tunnel (in networking terms) between the WSDB and the packager.

Examples of potential gateways include a nearby WSD with its own gateway capability, a smartphone running relay software (see Section III-C), a WiFi router, and a cell tower.

Although it may be built on complex lower-level protocols, a gateway actually provides only a very generic service: the relaying of (for example) IP packets. Database access through a gateway is a high-level protocol and there is no need to certify any of the lower-level protocols on which it is built.

E. Packager



The packager is a typically-software middle-ware that will sit between the client and the other two components (the gateway and the localization service(s)). It is responsible for translating requests, aggregating information, and providing a context for the information. Although the FCC and Ofcom mandate that the packager be part of the master², we expect that it will generally be found on the client because it is so lightweight.

The packager provides the following functionality:

- Receive information from localization services
- Create a message from localization info. and device ID
- Communicate with the client and gateway

Note that the packager is not *required* to interoperate with all localization services. Utilizing more services increases the quality of service for the client. However, in the worst case scenario when it cannot find or communicate with any localization services, it simply tells the client that there are no whitespace channels available at this time.

²According to the current regulations and consultations, the slave transmits its device ID to the master and receives a list of available channels in return. By definition the master is taking the role of packager in this case.

F. Interaction of components

The interaction of the three components is shown in Figure 2. The steps, numbered identically in the figure, are as follows:

- 1) The client sends its device ID to the packager. The device ID is already required by the TV WSDB access protocol [7, §15.711(b)(3)(iv)(A)] and we expect similar requirements in all shared-spectrum bands.
- 2) The packager optionally sends a request for localization information to a localization service. The necessity of this depends on the type of localization service used (e.g. broadcast vs. paired).
- 3) Localization information is digitally signed by the localization service and sent to the packager.
- 4) The packager assembles an access request including (multiple pieces of) localization information and the client's device ID. The request is encrypted and sent to the gateway.
- 5) The gateway forwards the encrypted access request on behalf of the packager. The whitespace database determines potential operating parameters for the client (e.g. a list of available channels), encrypts them, and sends them back to the gateway.
- 6) The gateway forwards the encrypted access response to the packager.
- 7) The packager forwards the encrypted access response to the client.

III. EXAMPLES OF THE SEPARATION OF RESPONSIBILITY

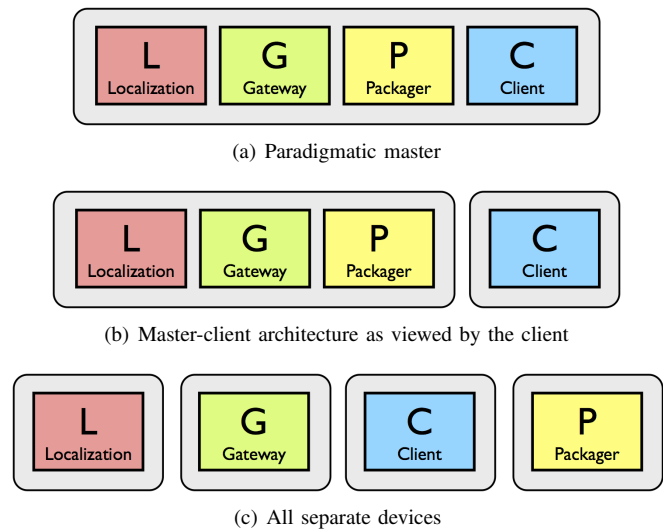


Fig. 3. Different configurations for the components of a whitespace device support network. Colored boxes represent components, whereas gray boxes denote a device containing one or more of these components.

So far, we have provided a description of the modular architecture that we are proposing. Under this proposed architecture, modules could be physically combined in a variety of ways, as shown in Figure 3. This section demonstrates the power of this proposed architecture via a series of real-world examples. Most will take the form in Figure 3(c) but the important point is that their form is unconstrained.

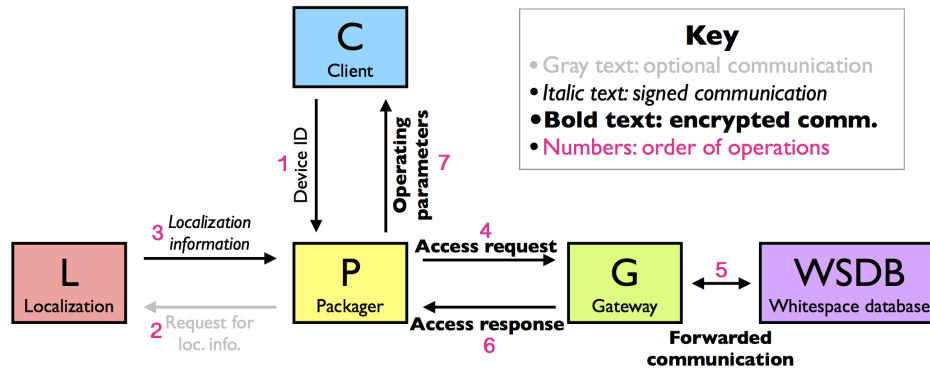


Fig. 2. Interaction between components of our proposed whitespace architecture (see Sections II-F for details).

For the sake of brevity and clarity, we will assume that the client contains the packager in all but the first example below. We believe this will often be the case.

A. Example 1: the current master-client architecture

As shown in Figure 3(b), the current master-client architecture naturally fits within our new framework. The gateway, localization service, and packager are contained within the master device, while the client device is bare-bones.

B. Example 2: gateway for bootstrapping

Even the same physical equipment may take on several of these forms throughout its lifecycle. In the UK, slave WSDs may have geolocation capabilities but no gateway. The master continually broadcasts very restrictive³ *generic* operating parameters to all slave devices in its service area. *If* these parameters allow for use of whitespaces, the slave may contact the master, providing its location in exchange for more permissive *specific* operating parameters. If not, there is no whitespace-based way for the devices to communicate.

In practice, it is difficult to obtain whitespace access via generic operating parameters alone due to their overly-restrictive nature. A proposed solution is the addition of a temporary gateway service (e.g. a nearby WiFi router or cell phone) for a geolocated slave device which allows it direct access to the specific operating parameters.

C. Example 3: smartphone as “master”

In this example, a smartphone running a regulator-certified application uses its built-in GPS to determine its own location. It pairs with the client device using Bluetooth, which limits the distance between master and client significantly. Thus the client’s uncertainty region is roughly a 100-meter circle centered at the smartphone’s GPS-determined location.

Using the smartphone’s data connection, the client establishes secure communications with the WSDB and sends its uncertainty region along with other identifying information.

³ To protect the primary, the restrictiveness of the generic operating parameters increases with the service area of the master. This is because the regulations allow for operation only on channels which are available in the master’s entire service area. The current form of the regulations means that often there is no whitespace available when using generic operating parameters.

In response, the WSDB securely replies to the client (through the smartphone) with a list of channels that are safe for the client to use.

The importance of this example relies upon the ubiquity of smartphones with data connections. If the WSD is expected to be used by a person, it may be reasonable for a manufacturer to require customers to use a smartphone app with their device rather than spend the extra money required to give it its own geolocation and gateway capabilities.

Another critical feature of this example is that the smartphone is not itself a whitespace device. Supporting devices could be used with a variety of spectrum-sharing devices, not only those in the TV whitespaces. Thus the infrastructure costs can be amortized over many devices instead of requiring a new infrastructure for each set of whitespaces. This idea of infrastructure reuse is hinted at in the PCAST report [4].

D. Example 4A: city-wide localization equipment

A city, wishing to provide services for its citizens and businesses alike, installs a beacon in a prominent location. This beacon is a fixed device, its location is certified by a technician, and it regularly broadcasts its location. This transmission may occur on any band, e.g. an open whitespace channel, an ISM band, or a cellular band. Any WSD that can hear the beacon is able to use its localization information to help determine its uncertainty region.

There are two important concepts in this example:

- The beacon may actually be a very cheap and simple device: it need not include a GPS unit nor does it even need an Internet connection. In principle, this beacon may be a simple box with an antenna which requires only (1) location-certified installation and (2) a power source.
- The load on the beacon is constant, regardless of the number of whitespace devices which benefit from its transmission. This is clearly preferable to the $O(N)$ scaling demanded by the existing master-client architecture.

E. Example 4B: city-wide localization equipment combined with building-level localization equipment

This example builds on the previous example in that it assumes the existence of a city-wide localization beacon. However, here we also consider a building manager who

has decided to enhance the whitespace access capabilities of devices inside his building by placing a second type of localization equipment on the roof of his building. This second piece of equipment has two advantages:

- In the event that the city beacon's signal is too attenuated indoors, it can act as a secure relay.
- If the equipment has additional sensing capabilities, it may be able to use techniques such as location fingerprinting (e.g. with WiFi or with TV signals themselves—see [6] for more details on how this could be done) to narrow down the uncertainty region.

This additional equipment may be a portable device which only needs access to electricity (perhaps via solar cells). Its location will not be certified which saves the building manager money by avoiding professional installation. The installation would ideally be as difficult as installing a clock.

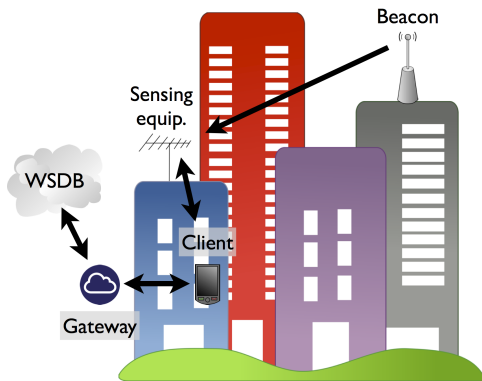


Fig. 4. Potential whitespace support network for indoor client

This scenario is illustrated in Figure 4. Clients within the building communicate with the sensing equipment. Time-of-flight information may be used by the building's equipment to certify that the client is within a small distance of the building's equipment. The clients then communicate the following information to the WSDB through a gateway:

- The beacon's location information collected by the building's equipment.
- Certification from the building's equipment that the client is within some range R_2 .
- (Optional) Additional localization information (e.g. TV signal levels) collected by the sensing equipment.

In this scenario, the WSDB would calculate the potential locations of the sensing equipment based on the beacon's location. In Figure 5, the beacon is at location known L_1 and the green dotted circle represents the possible locations of the building's equipment. The database then adds a buffer of size R_2 to the green circle to account for the client's uncertainty relative to the building's equipment. The construction is shown via the light blue solid-line circles (dark blue, L_2 , indicates the true location of the building's equipment). A list of channels safe for use in the final uncertainty region (shown in hatched purple) is computed by the WSDB and returned to the client.

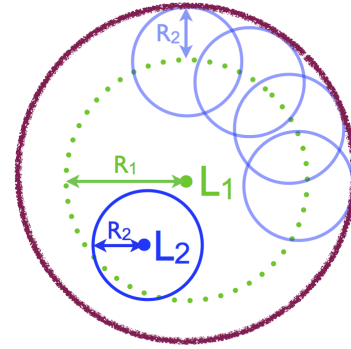


Fig. 5. Illustration of a safe way to chain localization information.

This example shows the potential for full separation of roles in order to inexpensively solve the problem of indoor whitespace access. Breaking out these roles grants innovators greater flexibility in the design of whitespace networks.

IV. CONNECTION TO SOFTWARE ENGINEERING PRINCIPLES

In this section, we will show how our proposed architecture satisfies the modern software engineering principles of modularity, testability, scalability, and upgradability. Each of these principles is well-recognized not only within the field of software engineering but also in many other engineering fields, especially design for manufacturing and assembly.

A. Modularity

Modularity is a design paradigm which stresses the separation of responsibilities among components of a system. A modular system is comprised of components, each of which has a specific task and a well-defined interface to the rest of the system. It is possible, in principle, to replace each component without needing to redesign the rest of the system.

Within the field of computer science, the benefits of modularity are widely known. As Liskov says in her book:

Modularity is the key to writing good programs. It is essential to break up a program into small modules, each of which interacts with the others through a narrow, well-defined interface. With modularity, an error in one part of a program can be corrected without having to consider all the rest of the code, and a part of the program can be understood without having to understand the entire thing. Without modularity, a program is a large collection of intricately interrelated parts. It is difficult to comprehend and modify such a program, and also difficult to get it to work correctly. [8]

The advantages of modular design have been long recognized in fields outside of computer science and electrical engineering. For example, the use of interchangeable parts was critical to the efficiency and success of assembly lines and other mass-manufacturing processes. The flexibility of being able to seamlessly substitute a new part for a broken

one greatly decreased costs since now items could be repaired instead of replaced. In the case of whitespace devices, modularity not only helps with testing but also with giving devices greater flexibility. By not rigidly defining the location or owner of the modules, many more designs are possible.

B. Testability

One of the great benefits of modular design is testability. While it can be difficult to certify that an entire system is working correctly due to its inherent complexity, it is often easier to certify each of its basic components individually. After all, the components are by definition simple pieces with well-defined behavior.

For example, one wouldn't dream of certifying a whitespace device and a whitespace database together as one unit, even though they will operate together in the field. Instead, regulators should test each individually to certify that it is operating with the defined parameters.

Using modularity to improve testability is a common software engineering practice. Liskov's book separates testing into two components:

Testing typically occurs in two phases. During *unit testing*, we attempt to convince ourselves that each individual module functions properly in isolation. During *integration testing*, we attempt to convince ourselves that when all the modules are put together, the entire program functions properly. [8]

What we are proposing is to apply this methodology to whitespace certification. Specifically, we propose that the components we identified be subject to unit tests.

One important consequence of performing unit tests rather than integration tests (as is currently the standard) is that now individual components can be patched or upgraded without requiring recertification of the entire system.

C. Scalability

As the number of networked devices increases at a dizzying rate, there is increasing emphasis on scalability in software design. Scalability refers to the ability of a system to handle additional requests (clients) without excessive overhead.

The paradigmatic master's load scales poorly with the number of clients, incentivizing master devices to conserve their resources only for clients within their system. Reducing this burden means that manufacturers will be much more likely to build interoperable systems with reciprocal infrastructure-sharing agreements. In our proposed architecture, the localization equipment can operate with a fixed cost regardless of the number of users via broadcasting its information.

Looking forward, we and the PCAST envision a future in which the TV whitespaces are only one set among many whitespaces [4]. In this future, localization services and gateways could serve the same purpose for all types of WSDs with no modifications. As WSDs increase in popularity, the number of localization services is likely to increase. This will generally mean a reduction in the size of uncertainty regions, therefore increasing the amount of recoverable whitespace.

D. Upgradability

In some sense, the current architecture is already remarkably modifiable, especially in contrast with previous spectrum allocation paradigms. For example, the following values specified in the regulations can be updated: coverage areas of primaries, separation distances, and power limits. Notice that all of these items are upgradable because the necessary data and computations reside in the *whitespace database*. The power of this model is shown by the rising popularity of software-as-a-service which enjoys very fast upgrade cycles since the software is hosted in the cloud.

Even in situations where providing a master device for the clients is not out of the question, clients will only see a performance boost when their specific master is upgraded. Under our proposed architecture, clients can more easily take advantage of upgraded hardware because of the greater likelihood for interoperability. Additionally, rather than needing to upgrade an entire master device, modules can be upgraded individually which makes incremental upgrades much more feasible.

V. CONSEQUENCES FOR CERTIFICATION

One of the goals of our proposed architecture is to simplify and robustify the certification process for whitespace services. Due to the modularity of the system, components can be certified individually rather than as a combined product. This allows for simpler per-module tests which are easier to carry out and inspire more confidence. In this section we describe the aspects of each component which must be certified.

Streamlining and standardizing the certification process will decrease the cost (in dollars and time) of certification. This is much more attractive to manufacturers who may otherwise fear uncertain and lengthy procedures.

Perhaps most importantly, the certification is no longer a matter of interference. Naturally, all three components must work in concert to ensure that the client does not operate illegally. However, the nature of the services they provide has nothing to do with interference. This makes the entire process less susceptible to concerns revolving around the vague term "harmful interference."

A. Certifying localization services

As mentioned previously and in Figure 1, the localization service has the following responsibilities that must be certified:

- *Determine absolute location (if applicable)*. Existing devices such as GPS navigators are already certified (e.g. by the FAA for aircraft navigation) so a similar certification process can be developed. Furthermore, absolute localization is already required by the FCC for whitespace devices so no additional challenge is presented.
- *Determine bounds on distance to another device for chaining of localization information (if applicable)*. This would be needed in systems such as the one in Example 4B. Again there are existing systems (e.g. the E-911 system) which provide such distance estimates.

It is evident many of the pieces in the localization service already have a candidate certification process. Those which do

not are still easy to test. It would even be possible to create open-source government-blessed software packages which perform these tasks. Since the problem is no longer at all related to “harmful interference,” the standard for such software could be reasonably outsourced to an organization such as NIST (National Institute of Standards and Technology).

B. Not certifying the gateway

The worst thing a gateway can do is fail to relay data. We assume that the traffic being routed through the gateway is appropriately encrypted, which prevents eavesdropping and tampering. Furthermore, we suggest a scheme that prevents replay attacks, man-in-the-middle attacks, etc. The bottom line is this: *regulatory bodies need not trust the gateway.*

C. Certifying the packager

As with the localization service, the packager needs to be certified to create and encrypt messages containing the correct information (localization information, device ID). This can be tested in a similar manner via software unit tests.

D. Certifying the client

The pieces of the client requiring certification are already supposed to be certified by regulators under the existing architecture. For example, the client must already be certified to provide the correct device ID and to operate according to specified parameters. Thus no changes are needed here.

E. Not certifying the system

Certifying that each piece of the system operates correctly frees the regulator to think separately about how all of the pieces will interact. Regulators can create models and simulations of these components to examine their interactions in order to prevent any unfortunate combinations *before* they show up in an actual system. Using simulations allows regulators to carry out much more thorough and reassuring studies than could ever be done on a per-device or per-system basis.

Even in the event of a failure, the regulator maintains a high degree of control over the system via the databases. In most cases, we expect that system-wide flaws (e.g. an unforeseen interaction of components) would require only an update to the WSDB code. In the worst case, devices or components can simply be denied access to the whitespaces, either temporarily or permanently. This ability to control even deployed systems inspires confidence that WSDs will operate correctly.

VI. CONSEQUENCES

A. Economic incentives

With this new perspective on the minimal requirements for the whitespace access problem, we see that *existing* devices such as smartphones are only a software upgrade away from being whitespace-enabling devices. By building on top of existing technology, our proposed architecture enables accelerated growth within the WSD market. Client devices—which are much cheaper than the paradigmatic master devices and have the added benefit of actually working indoors—are now truly within reach. The bottom line is this: *there is an economic incentive to adopting this potential separation of responsibility.*

B. Lighter regulations

Although our proposal has major beneficial consequences for devices and device manufacturers, the regulatory changes needed are actually minimal. With the exception of the chaining of localization information—which is optional and need not be implemented in the first round of regulations—our proposal essentially requires more general wording within the regulations but no new ideas. We wish to emphasize that our proposal is actually for rules which are *lighter* than the existing regulations, not more burdensome.

C. Market penetration

As the number of whitespace-related devices increases, the probability of being unable to access any particular component of a whitespace device support network decreases. Thus as the market penetration increases, devices will be able to more precisely locate themselves and thus recover more whitespaces. Until infrastructure is widely available, devices may use cheap techniques with poor-but-usable service and plan to opportunistically use current and future infrastructure.

VII. CONCLUSIONS

We have proposed a modular approach to the design and regulation of spectrum-sharing devices. One of our key observations was that although the components identified above need to be certified for compliance with whitespace rules, *they do not need to be part of a whitespace device.* We also argued that the proposed separation of responsibilities allows for more reliable testing and certification. Finally, we described the economic and regulatory benefits of our proposed architecture.

VIII. ACKNOWLEDGEMENTS

We thank the U.S. NSF (CNS-1343398, CNS-1321155, and AST-1444078).

REFERENCES

- [1] “In the Matter of Unlicensed Operation in the TV Broadcast Bands: Second Report and Order and Memorandum Opinion and Order,” Federal Communications Commission, Tech. Rep. 08-260, Nov. 2008. [Online]. Available: http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-08-260A1.pdf
- [2] K. Harrison, S. Mishra, and A. Sahai, “How much white-space capacity is there?” in *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*. IEEE, 2010, pp. 1–10.
- [3] “TV white spaces: A consultation on white space device requirements,” Nov. 2012. [Online]. Available: <http://stakeholders.ofcom.org.uk/binaries/consultations/whitespaces/summary/condoc.pdf>
- [4] President’s Council of Advisors on Science and Technology, “Realizing the full potential of government-held spectrum to spur economic growth,” Tech. Rep., 2012. [Online]. Available: http://www.whitehouse.gov/sites/default/files/microsites/ostp/pcast_spectrum_report_final_july_20_2012.pdf
- [5] S. Mishra and A. Sahai, “How much white space has the FCC opened up?” *IEEE Communication Letters*, 2010.
- [6] K. Harrison and A. Sahai, “Allowing sensing as a supplement: an approach to the weakly-localized whitespace device problem,” in *Dynamic Spectrum Access Networks (DYSPAN), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 113–124.
- [7] “In the Matter of Unlicensed Operation in the TV Broadcast Bands: Third Memorandum Opinion and Order,” Federal Communications Commission, Tech. Rep. 12-36A1, Aug. 2012. [Online]. Available: http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-12-36A1.pdf
- [8] B. Liskov and J. Guttag, *Program development in JAVA: abstraction, specification, and object-oriented design*. Pearson Education, 2000.