Article

Towards Collaborative and Dynamic Spectrum Sharing via Interpretation of Spectrum Access Policies

Jakub Moskal[1], Jae-Kark Choi [2], Mieczyslaw M. Kokar [1,3], Soobin Um[4], Jeung Won Choi[5]

1 VIStology, Inc.; jmoskal@vistology.com
2 Hanwha Systems Co. Ltd.; jaekark.choi@hanwha.com
3 Northeastern University; mkokar@ece.neu.edu
4 Agency for Defense Development; friedman61@kaist.ac.kr
5 Agency for Defense Development; jwchoi@add.re.kr
* Correspondence: mkokar@vistology.com

Abstract: This paper describes some of the challenges that need to be addressed in order to develop collaborative spectrum sharing systems. The importance of these challenges stems from the assumption that rules for spectrum sharing can change after the deployment of radio networks and the whole system must be able to adapt to them. To address such a requirement, we used a policy-based approach in which transmissions are controlled by a policy interpreter system, and the policies can be modified during system operation. Our primary goal was to develop a prototype of such a system. In this paper, we outline the implementation of policy interpretation, automatic generation of transmission opportunities in case a request for transmission is denied by the policy reasoner, and the generation of rendezvous channels for the synchronization of otherwise asynchronously running software defined radios.

Keywords: spectrum sharing, collaborative spectrum sharing, policy-based sharing, transmission opportunities, rendezvous channels, policy interpreter

1. Introduction

A number of definitions of *cognitive radio* have been published in the literature. We are using here a definition that is provided by the IEEE P1900.1 standard [1]:

(a) A type of radio in which communication systems are aware of their environment and internal state and can make decisions about their radio operating behavior based on that information and predefined objectives.
(b) Cognitive radio [as defined in item (a)] that uses software-defined radio, adaptive radio, and other technologies to adjust automatically its behavior or operations to achieve desired objectives.

To realize the full potential of this definition, a number of capabilities need to be supported by a radio [2]. Among others, the capabilities should include: (1) Sensing of the RF environment and information collection (*meters*); (2) Answering user's queries as well as of other cognitive radios; (3) Situational awareness – not just knowing the state of the RF environment, but also understanding the various relations between the parameters; (4) Self-awareness – knowing own parameters and settings (aka *knobs*); (4) Issue queries/request to other radios; (5) Command execution – accepting requests from other radios, making decisions and implementation of commands; (6) Accessing and interpreting knowledge bases and data bases that contain information relevant to the control of its own capabilities (adaptation).

Figure 1 shows the context for cognitive radio. Two radios shown in the figure support APIs to User, Sensor, Knowledge Base and other radios within a Radio Network. Since in this paper we are primarily focusing on the dynamic spectrum sharing (DSS) among the radios operating in the same RF environment – some in the same Radio Network and some outsiders – we need to cover the issues of collaboration. Collaboration is defined as the process of two or more entities or organizations working together to complete a task or achieve a goal. Collaboration is an inherent part of communications. Thus, the question is

check for updates

Citation: Moskal, J. J..; Choi, J-K.; Kokar, Mieczyslaw, M. M.; Um, S.; Choi, J. W. Dynamic Spectrum Sharing. *Preprints* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

how to implement collaboration rather than *whether* to support it. Collaboration implies the willingness of the entities to interact and make trade-offs with each other. The organization of the collaboration process must follow some policies that all the entities should obey. In the use cases considered in this paper, the objective of collaboration is to make collaborative decision about the use of the RF spectrum - a crucial step in Electromagnetic Spectrum Management (ESM).

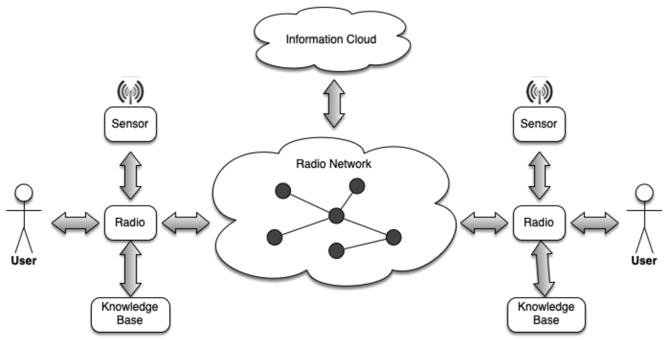

Figure 1. Cognitive radio in context

The next question is how policies should be encoded on a radio? The most typical approach is to encode each policy in an imperative language, e.g., C++, and then implement a control loop that matches current ESM situations to the existing policies and executes the matched policies. The advantage of this approach is that policies are hand-coded by programmers and thus their execution will most likely be computationally efficient. However, this approach also has a number of disadvantages. The most difficult requirement to satisfy with this approach is the possibility of modification of the code after deployment. Whenever a need arises to modify a policy, new procedural code needs to be developed, compiled and deployed.

Another approach is to add a declarative layer (a meta-layer) of code, to the imperative code. E.g., it can be an XML file that provides pre-conditions for a policy execution as well as the postconditions that need to be satisfied by the controller. While this approach is more flexible than the one just described above, it still requires re-coding of the testing of the pre-conditions as well as the procedures implementing the post-conditions.

The cognitive radio requires that the language in which policies are coded must be machine processable and understandable. In other words, the language must be a formal declarative language with formal syntax and semantics. There are two kinds of knowledge that needs to be represented using such a formal declarative language: (1) the shared concepts between radios and networks; (2) the policies that are used to control the behavior of the radio.

The shared concepts between radios and network can be defined in common ontologies. In artificial intelligence, computer science and information science, ontology means a formal, explicit specification of a set of concepts in a specific domain and the relationships between these concepts. The term formal means that the ontology is machine processable for the purpose of knowledge reuse and sharing [3].

2. Cogntivie Policy Based Cognitive Radio

According to [1], *Hardware Radio* is a type of radio whose all communications functions are entirely implemented in hardware, but most importantly, changes in communications capabilities can only be achieved by changing the hardware. Software Defined Radio (SDR) is a radio in which some or all of the physical layer functions are software defined, i.e., implemented in software (which implies more than just controlling the functions implemented in hardware). *Policy Based Radio (PBR)* is a type of radio whose behavior is governed by sets of rules, expressed in a machine-readable format. Although PBRs are usually SDRs, this is not a logical requirement for being policy-based. *Cognitive Policy Based Radio (CPBR)* is a type of radio that is aware of its environment and internal state and can make decisions about its radio operating behavior based on that information and predefined objectives. Access to the Electromagnetic Spectrum (EMS) is one of the aspects that is controlled according to policies it has. In most cases, this means that they dynamically sense the spectrum for current usage and use the parts of it that are not occupied by the Primary Users (PU). As soon as a PU is sensed and there is evidence that transmitting in the PU's band could result in interference to the PU, the CR vacates the current channel and tries to grab a different, unoccupied part of the spectrum. The decisions on whether to use a specific spectrum can be made via the interchange of messages between the collaborating radios, or it can be implemented autonomously by each node following the locally available policies.

Policies specify a number of factors that should be taken into consideration by the radios in order to arrive at a spectrum use decision. E.g., the location of the device, the frequency and time range of interest, or the maximum transmit power. Different policies pertain to different geographical areas and can change dynamically. Therefore, the CR design cannot account for all possible policies and must incorporate a mechanism for interaction with some authority that can dynamically grant or deny access to particular parts of the spectrum. In our approach, we have a *Policy Manager (PM)* component to interact with such an authority. Also, in our approach, a Policy Engine (PE) is present on the radio and is used to ensure that the radio operates within the constraints established by the PM. The PM updates its policies based on the geolocation of the device by wirelessly interacting with an external entity when entering a new region.

The interaction inside the radio between the main Controller and the PE requires a language for expressing the spectrum access requests and the responses to such requests. In case the access cannot be granted, the PE may respond with a list of opportunities, i.e. other channels, in which opportunistic access is possible. The above requirements cannot be achieved via static definitions of vocabularies (e.g., in XML or a relational database), since adding new terms to such vocabularies would require developing new software (for interpreting XML tags or relational schemas and data). On the contrary, this kind of capability can be achieved by an ontological approach in which new terms and their definitions can be introduced dynamically. This is the consequence of bounding ontologies to a highly expressive language (like OWL [4]) in which new terms can be defined on the fly. These definitions are then interpreted by generic *inference engines* that are bound to the same language (like OWL), thus avoiding the need to rewrite the underlying software.

While opportunistic access to the RF spectrum is reasonable for addressing the problem of no-interference to PUs, such an approach does not solve the problem of *spectrum sharing* among the radios within the same group, i.e., either PUs or the friendly secondary users (SUs). In this paper, we present parts of a policy-based solution to such cases. First, we briefly describe an architecture in which such scenarios can be investigated. Then we describe our experiments with policy interpretation, policy-based generation of transmission opportunities, policy-based definition of rendezvous, and policy-based optimization of radio parameters (knobs) based on the transmission feedback (meters). Finally, we analyze the consequences of not having some of the capabilities considered in this paper.

3. Architecture

The main component of Cognitive Radio is a Cognitive Engine, as shown in Figure 2. The architecture of the radio must support the functionality described above, plus it must support the aspects of policy management and policy interpretation and execution. Most of the cognitive system architectures are organized around an iterative processing loop (cf. SOAR [5], ACT-R [6]). Some others, e.g., the Endsley's model [7], are patterned after the OODA loop - Object, Orient, Decide and Act [8]. The OODA loop in general, and Endsley's model in particular, have been widely used in the domain of Information Fusion, where it serves as a basis for situational awareness.

Mitola [9] proposed to extend the OODA loop and use it as the basic processing loop of SDR-based cognitive radio. He introduced two more processes to the control loop - Plan

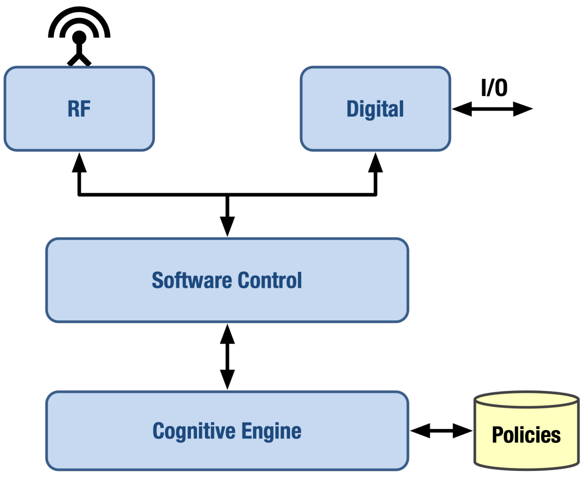

Figure 2. Cognitive Radio - top level

and Learn. This model, also known as ideal CR architecture (iCRA), is characterized by the Observe-Orient-Plan-Decide-Learn-Act (OOPDLA) loop (see Figure 3). The processes inherited from the original OODA model have the following interpretation in the context of cognitive radio: *Observe* - collect data about the environment using an array of sensors; *Orient* - integrate the observations with policies, goals and preferences, hardware and software limitations and past experience; *Decide* - identify the changes that need to be made in the SDR configuration in order to address the new situation; *Act* - reconfigure the SDR. And the added processes are: *Plan* - planning decisions for the long run; *Learn* - machine learning and adaptation based on past experience.

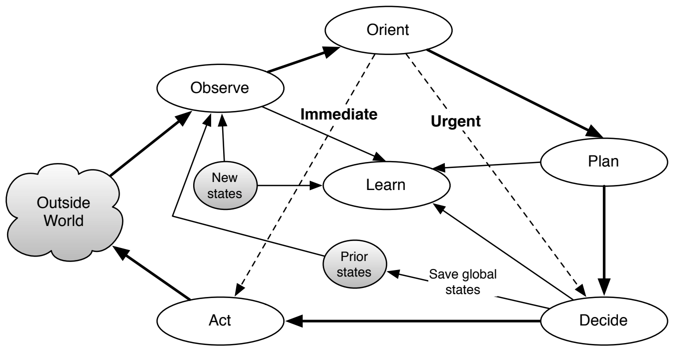

Figure 3. OODA Loop extended

One of the possible passes through the OODA loop is as follows:

1. Receive a request (from Software Control) to transmit a packet of information; the request includes the metadata - frequency, time, power, and other items of information, as needed.
2. Identify policies relevant to the current request.
3. Invoke policy engine (PE) to verify that the transmission request is compliant with at least one of the policies and not forbidden by any of the relevant policies.
4. If the decision is "allow", then convey this decision to Software Control.
5. If the decision is "disallow", then invoke the function Compute Transmission Opportunities (planning).
6. If transmission opportunities found then send them to Software Control; otherwise send the "disallow" decision
7. Continue to point 1.

The architecture of the experimental system used in this work - termed Cognitive Radio Engine (CRE) - is shown in Figure 4. It consists of six components: Controller, Policy

Table 1: Mapping OODA to Architecture

OODA	Function	Component
Observe	Get/describe transmit request	Controller
Orient	Identify matching active policies	Policy manager
Observe	Get spectrum measurement	Controller
Decide	Verify compliance	Policy Engine
Plan	Generate transmission opportunities	Planner
	Schedule rendezvous	
Act	Issue transmit command	Controller
Learn	Update policy	Learner

Manager, Policy Engine, Planner, Learner and Inference Store. Most of the interactions of the CRE are coordinated by its Controller component using the specific component dedicated APIs. The components of this architecture are in a direct relation to the OODA loop as shown in Table 1. The table shows some functions that have not been discussed so far, but are needed for the realization of collaboration among the CRs. In particular, we identify the operation of Schedule Rendezvous as a functionality that is needed for synchronizing collaboration among the radios.

Most of the functions invoke the Inference Store component. The information in the CRE is represented in OWL (Web Ontology Language [4]) and an OWL Reasoner is part of the Inference Store. The functionality of the particular components is described in more detail in the rest of the paper.

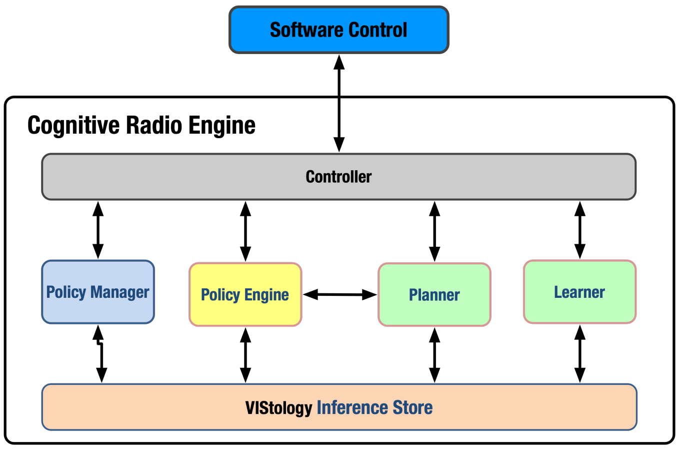

Figure 4. Cognitive Radio Engine Architecture

4. Ontology

For the implementation of communications systems two parts are needed — a protocol to exchange messages and a language in which the messages are represented. In our work, we make use of *ontologies* as a representation language. First, we briefly introduce the notion of ontology since this concept is relatively new to the radio communications community. To put the notion of ontology in perspective, we overview other concepts that are close to ontologies.

Vocabulary: It is a listing of terms, e.g. Radio, Receiver, Transmitter, Antenna, Battery. All it contains is lexical representations of different concepts that a human reader can interpret just because the reader knows these names. A computer program would threat them simply as strings and thus would be able to perform operations that are applicable to the data type String, like compare, insert, delete, assess the length of the name and such.

Dictionary: It includes terms with natural language explanations of their meaning, where the explanations use the same language that includes the defined terms. Here is an example of dictionary term "radio" as can be found by Google if you enter the search term "radio".
Radio: Noun: *the transmission and reception of electromagnetic waves of radio frequency, especially those carrying sound messages*. Use: "*cellular phones are linked by radio rather than wires*"; Verb: *communicate or send a message by radio*. Use: "*the pilot radioed for help*".

Dictionary contains more information than vocabulary since it can be used by the human to understand what a term means in case the human does not know it. However, dictionaries are difficult to process by computers since computers are not too good (so far) with processing natural language text.

Ontology: An ontology includes a number of *classes*. E.g., Radio can be one of the classes. Any class can have multiple *instances* (aka *individuals*) and thus the Radio class can have a number of individuals - members of the Radio class, e.g., R-AFK2178, RD-FM-365M, and such. An ontology can represent the facts that Radio comprises a Transmitter, a Receiver, an Antenna, a Speaker, a Frequency Selector, a Modulator, a Demodulator and such. This would be encoded in the ontology (conceptually) as triples $< subject, predicate, object >$. Here subject would need to be an individual from a class, while object is either an individual of a class, or an instance of a data type, e.g., Integer or String. All individuals belong to at least one class called Thing (the top class). The classes in an ontology are organized along the subClassOf relation, resulting in a hierarchy (a lattice). Thus any individual from any class is a thing (via a chain of the subClassOf relations).

Predicate is an individual of a special class called Property. Properties represent relations between pairs of individuals. E.g., the fact that a specific radio has a transmitter would be represented as a triple $< R - AFK2178, hasPart, TR - FRK0999 >$. Here hasPart is a predicate. The facts that R-AFK2178 is a radio and TR-FRK0999 is a transmitter would be represented as two other triples $< R - AFK2178, rdf : type, Radio >$ and $< TR - FRK0999, rdf : type, Transmitter >$, in which a special OWL predicate *rdf:type* is used to associate individuals with classes.

Predicates represent either relations between individuals, or attributes of individuals. Some of the characteristics are the "defining characteristics". If an individual has those characteristics, it can be classified as an instance of that class. E.g., a thing that has a transmitter and a receiver must be an instance of the class Radio.

Some other characteristics are just "descriptive", e.g., characteristics that are common to various classes (various things have the property of weight). If it is known that a thing is an instance of a given class, then it is expected to have characteristics that are necessary of such a class. E.g., if a thing is an instance of the class Radio, then it must have a transmitter and a receiver.

For ontologies to be processable by computers they must be described in a language that has formal semantics. To be "understandable", the language must also have formal, computer processable *semantics*. Roughly, it means that there needs to be a computer program that can answer queries about information encoded in the language. E.g., in the case of databases, the query can be expressed in SQL and a database management system can provide answers - retrieve data stored in the database. The answers need to be *sound*, i.e., the answers must be correct with respect to what the database contains. For an ontological language we require more than a database. An ontological language must have a program called "*reasoner* (or *inference engine*, which can do more than just retrieve information from its *knowledgebase*, but it should be able to extract information that is only implicit in the knowledgebase. E.g., if the knowledge base describes the radio communications domain, it will encode facts about some of the domain objects as *axioms* expressed in the language. The reasoner then will be able to process the input facts and the axioms and answer queries based only on partial information about an object. E.g., if the knowledgebase contains an object classified as jammer, the reasoner will be able infer that it has a transmitter. The use of axioms has very powerful consequences - it provides means for extracting facts that are only implicit and for checking the logical consistency (lack of contradictions) of the knowledgebase.

4.1. A DSA Ontology

Since our approach heavily relies on an ontology, we first briefly overview the ontology used in this paper.

Ontologies, similarly to other engineering artifacts, like hardware or software, are built in a modular and hierarchical fashion. The structure of the DSA ontology (we term it here OntoDSA) we use is shown in Figure 5. The arrows in this figure represent the "import" relationships. OntoDSA imports SpectralSPARQL, which in turn imports GeoSPARQL as well ontologies to describe Lists and Uncertainty (both of them import the Nuvio foundational ontology.

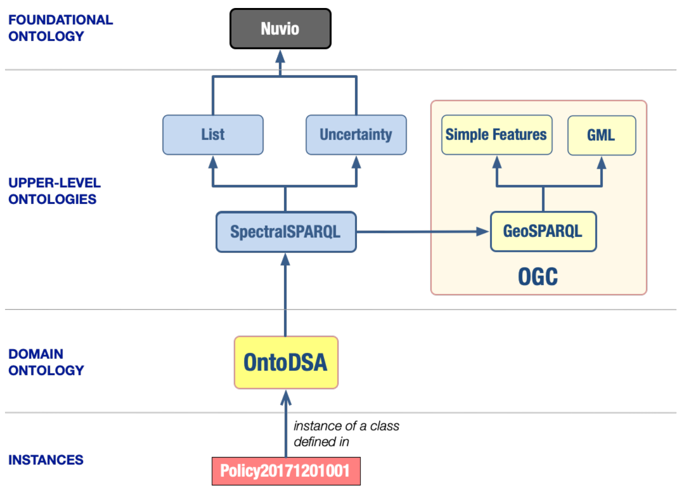

Figure 5. The Architecture of the DSA Ontology.

The structure of SpectralSPARQL is patterned upon GeoSPARQL [10] — an Open Geospatial Consortium (OGC) standard extension of SPARQL, which provides a common representation of geospatial data in RDF [11] (Resource Description Framework, a language for the Semantic Web), and facilitates querying and filtering of such data based on relationships between geospatial entities. The standard consists of an ontology of geospatial concepts, a set of SPARQL functions and query transformation rules. With GeoSPARQL, one can express high-level queries that ask about specific geographic objects of features, their attributes and relations between them.

SpectralSPARQL is built on top of the base ontology, Nuvio [12], and thus includes its top-level classes: Attribute, InformationEntity, Object, Process, Situation, UnitOfMeasure and Value; all mutually exclusive. Each of the classes in SpectralSPARQL are subclasses of at least one of these concepts. The two top-level concepts in Nuvio are Object and Process - entities that are either wholly represented, or that can only be partially represented at any given snapshot of time, respectively.

The three fundamental objects in SpectralSPARQL are Signal, Receiver, and Emitter. The process subclasses are used to model different activities that the objects can participate in. For instance, an emitter can participate in a frequency hopping transmission.

The subclasses of the Attribute class model properties for describing both objects and processes. Some of them are simple quantities with associated units of measure, e.g. Frequency with FrequencyUnit, while others are more complex and require special-purpose properties to fully define them. There are three major categories of Attribute - Spectral, Temporal and Signal. The first one is the central focus of the SpectralSPARQL functions. The SpectralAttribute class was modeled in a similar fashion as Feature in GeoSPARQL. It is used to define SpectralSPARQL functions that calculate relationships between two different spectral attributes, e.g. intersection, regardless whether they are single frequencies, frequency sets or ranges, much like the *Intersects* query function in GeoSPARQL.

The OntoDSA ontology imports all of the above by simply importing SpectralSPARQL. The most significant classes in this ontology include Policy, Rule, and the SAPHyperspace classes, and relate directly to the Spectrum Access Policy use cases implemented by the CRE:

- Each instance of the Policy class can be related to many instances of the Rule class by the property *hasRule*. It may also have several data property assertions, such as *hasID*, or *hasRevision*.
- Instances of the Rule class can be related to instances of the SAPHyperspace by the property *governs*, denoting a relationship between a SAP rule and the spatio-spectr-temporal region that it applies to. Properties of each rule such as its *ID, revision*, or *type* - which can be either *allow* or *deny*, or maximum transmit power are asserted directly on the instance of the Rule class.
- Instances of the SAPHyperspace class represent spatio-spectro-temporal regions that are governed by some SAP rules. The term *spatial* is interpreted here in a more abstract sense, more than just the physical space. Any of the quantities can be considered as a dimension of such a space. Each hyperspace defines its frequency range, time interval and the geospatial region, to which the associated rule is applied, and more. While the first two properties are defined using the SpectralSPARQL concepts, the latter is defined using GeoSPARQL.

An example of a policy is shown in Figure 7 later in the paper, after we discuss policies.

5. Policies

We distinguish two kinds of policies: (a) Hyper-cube based policies — constraints and requirements based on sets of states of the environment and the transmitter, and (b) Higher-level policies that are defined by logical conditions. First, describe each policy kind in greater detail, then we show an example of a logic-based policy.

5.1. Hyper-cube Based Policies

The policies are formalized by a number of rules, P_p. The rules are formally defined as functions that assign decisions depending on subsets of conditions, C, that are either satisfied or not. The conditions are expressed either as subsets of states, S, of the RF environment and radio characteristics, or logical conditions on such states. A generic form of a rule can be formalized as:

$$P_p : 2^C \to D \tag{1}$$

where the conditions C are combinations of set-based and logical conditions:

$$C = S \cup L \tag{2}$$

S is an explicitly specified set of states, and L is a set of logical values - expressions of the form $2^L \to \{T, F, U\}$, where T stands for logical *true*, F for *false* and U for *unknown*.

In a specific case, the sets can be intervals of dimensional values representing X and Y locations - X_i, Y_i, time intervals T_i, frequency ranges F_i, transmit power W_i and other types of sets. An example of a spectrum access policy signature is:

$$P_p : X_i \times Y_i \times T_i \times F_i \times W_i \times C_i \to \{a, d\} \tag{3}$$

where $\{a, d\}$ shows whether the policy is *allowed policy* or *restrictive policy*. P allows to transmit when the value is a, while disallows to transmit when it is d.

Examples of projections of this kind of policies are shown in Figure 6, where the policy C is a restrictive policy, while the rest are allowed policies. This figure shows projections from 6D to 2D. In 6D, we refer to them as hypercubes, or D-rectangles (in this case D = 6).

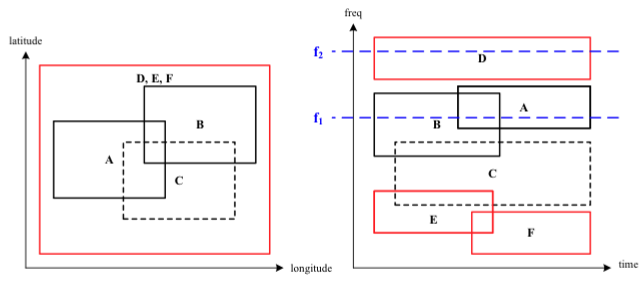

Figure 6. An Example of Policies.

5.2. *Higher-level Policies*

Policies that are based on logical conditions cannot be represented as hypercubes. We make use of the SPARQL query language for representing policies, in addition to its obvious use for querying. More specifically, in this paper we use a proposed query language for the domain of spectrum dependent systems, called *SpectralSPARQL*, described in [13].

As an example, consider a policy expressed in natural language:

> Allow transmission for a transmitter T located at least $2km$ from *NodeX* provided that T's Tx power is below P_1 and its MCS is QPSK or if T is located at least $3km$ from *NodeX* provided that its Tx power is below P_2 and its MCS is 16QAM.

This query can be expressed in SpectralSPARQL as shown in Listing 1.

```
INSERT {
  ?request :hasDecision :allow
}
WHERE {
  {
    ?request a :TransmissionRequest;
    :forNode: ?Node1 .
    ?Node1 a :Emitter;
    :hasMCS QPSK;
    :hasTxPower ?TxPower;
    :distanceToNodeX ?distance.
    FILTER (?distance > 2000 && TxPower < P1)
  }
  UNION
  {
    ?request a :TransmissionRequest;
    :forNode: ?Node1 .
    ?Node1 a Emitter;
    :hasMCS 16QAM;
    :hasTxPower ?TxPower;
    :distanceToNodeX ?distance.
    FILTER (?distance > 3000 && TxPower < P2)
  }
}
```

Listing 1: A SPARQL representation of the policy

6. An Implementation of a Policy Based Spectrum Sharing System

In this section we describe how the architecture of Figure 4 was implemented in our experimental system.

6.1. Controller and SDR Platform

As was shown in Figure 2, the radio platform includes two kinds of processing - RF and Digital. The RF part receives and transmits radio signals, which are processed in the Digital processing, which both receives information to be sent out and outputs the received and processed information to the radio users via its I/O streams. In case of SDR, the radio platform also has a Software Control module which is used for setting RF and Digital hardware and software, as well as controlling the radio behavior. The Software Control module interacts with the Controller module of the Cognitive Radio Engine through the Controller's API. This API handles both control messages, e.g., request for permission to transmit, and data messages, e.g., packets that CRE needs to send to other radios. This API is also used for getting policies from the Policy Authority, as needed.

One of the primary responsibilities of Controller is translation between the language used to capture information in Software Control and the language used in the CRE. Since on the CRE's side everything is expressed in OWL and SPARQL and on the Software Control side either in JSON or CORBA IDL, the Controller's APIs must support this "adaptation". Controller passes policies. It also allows the CRE to get *meters* and set *knobs* through which the radio platform is monitored and controlled.

6.2. Policy Manager

The PM manages policies and associated policy interpretation rules. In particular, it maintains a list of *active policies*, i.e., the policies that are applicable to the current circumstances as imposed by the Authoritative Policy Source. The function of PM is to support the authoring of new policies, editing of the existing policies, storage and retrieval of policies in policy database, activating/deactivating policies. For this purpose, Policy Manager supports functions that execute such tasks as Select Active Policies, Add Active Policy, Remove Active Policy, Select Allowed Policies, Select Disallowed Policies, and so on.

Looking deeper into the structure of the system, PM manages the policies that are stored locally in the Inference Store (see Section 6.3), based on actions initiated from the Controller. Each time a modification is made (rules are modified, policy is activated, etc.), Inference Store automatically invokes BaseVISor [14] - an OWL Reasoner - to infer any implicit facts about the policies. All facts, explicit and derived, are subject to querying when, for instance, PE is determining the transmission opportunities, which requires access to the currently active policy definition.

6.3. Policy Engine and Inference Store

Policies are interpreted by the PE component. The most important function is to infer whether a request for transmission can be granted or not. The case of "allow" is rather simple; the Controller returns a permission (allow) to Software Control, which then can start transmitting. In case it is a deny, there may be a bigger issue that should be resolved by PE. For instance, if the request cannot be granted for transmitting at this time and at this power, the question is — what should the Controller do? If the only response from PE is allow or deny, the Controller may need to attempt many different requests (assuming it knows how to generate them) before one is finally granted, if at all. To avoid this "long-loop" scenario, PE can produce a list of *transmission opportunities* that would be allowed in the current state and given the current policy.

Internally, to generate opportunities, Controller uses Planner (see Section 6.4) and VIStology's Inference Store. Inference Store (IS) is a component that combines the power of BaseVISor's OWL inference capability [14] with an in-memory triple store that exposes a SPARQL 1.1 interface [15]. Upon changes to the triple store (assertion of new facts), BaseVISor is automatically triggered and injects implicit facts into the store. Thus, at query time, all facts (explicitly asserted and inferred) are immediately available.

IS stores the currently active policy expressed in an ontological form and is managed by the PM. Each policy is represented in OWL according to the structure that is partially represented in Figure 7. The top-level class, SpectrumAccessPolicy can be marked as active and PM ensures that exactly one policy is active at any given time. The policy comprises a number of SpectrumAccessRules that are related to it via the *hasRule* object property. Each rule *governs* exactly one SAPHyperspace, which represents a multi-dimensional space defined by frequency range, time interval, a geospatial region and other sets, as described in Section 5. The concrete definitions of these dimensions are imported from pre-existing ontologies: SpectralSPARQL [13] and OGC GeoSPARQL [10], respectively. Finally, the rules can be of two types: Allowed or Denied, modeled as subclasses of the SpectrumAccessRule class. Instances of the Allowed rule can specify the maximum transmit power (defined in SpectralSPARQL) that can be used within the governed hyperspace. The ontology was designed using the best ontology engineering practices, i.e. modularity and reuse, which allowed for a rapid development and supports integration of the policy data with other sources of information in future scenarios.

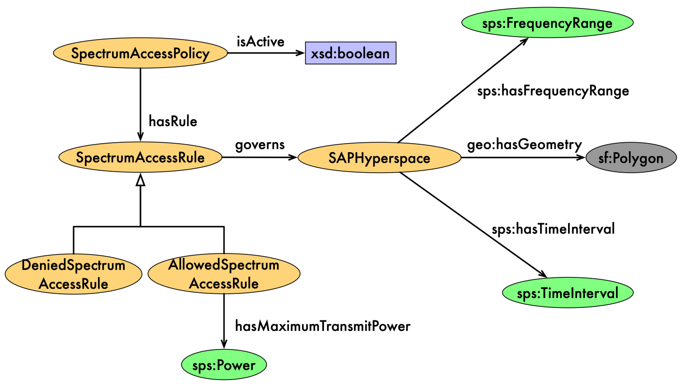

Figure 7. A snapshot of the SAP Ontology.

6.4. Planner

The responsibilities of the Planner module include, among others, generation of transmission opportunities (mentioned above) and generation plans for rendezvous. Other responsibilities may be added to this module as needed.

6.4.1. Transmission Opportunities

While computing opportunities may seem to be a simple issue, the problem is that the number of possible opportunities may be quite large, especially for policies that comprise large numbers of rules. One simplistic way to solve such a problem would be for the Controller to continuously generate requests (in a loop) in response to denies, this would mean very many interactions between the Controller and the PE; clearly not a great solution from the point of view of the efficiency of the CRE. In our approach, the PE provides an API to the Controller through which it can query the PE about the transmission opportunities for any location, time, frequency and power under the currently active policies.

In search for possibly readily available solutions, we reviewed a number of published algorithms. One of the algorithms that relatively well matched the requirements is called the *BSP — Binary Space Partitions* (cf. [16] [17]). After a closer look, we directed our search toward *k-d* trees (short for k-dimensional tree) [18], which is a space-partitioning data structure for organizing points in a k-dimensional space. *k-d* tree, a special case of a binary space partitioning tree, is a useful data structure in applications such as search, involving a multidimensional search key (e.g. range searches and nearest neighbor searches). We have also looked at the graph-theoretic concept called *boxicity*. In graph theory, boxicity is a graph invariant. The boxicity of a graph is the minimum dimension in which a given graph can be represented as an intersection graph of axis-parallel boxes. In other words, there must exist a one-to-one correspondence between the vertices of the graph and a set of boxes, such that two boxes intersect if and only if there is an edge connecting the corresponding vertices. Our conclusion here was that this concept is not directly applicable to our problem. Finally, we analyzed the *R-trees* [19] and *R*-trees* [20]. *R-tree* data structures are used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons. *R*-trees* are a variant of *R-trees* used for indexing spatial information. *R*-trees* have a slightly higher construction computational complexity cost than standard *R-trees*, as the data may need to be reinserted. However, the resulting tree usually has a better query performance. Similarly as the standard *R-tree*, it can store both point and spatial data.

In our implementation, both policy rules and multi-criteria transmission requests are viewed as hyper-rectangles, which is a simplification assumption for hyperspaces, stemming from the fact that rules approximate geographical areas as rectangles, as opposed to arbitrary polygons in a more general case. Edges in each hyper-rectangle dimension correspond to intervals along the geographical dimensions of longitude, latitude, and elevation, the time interval, the frequency interval, and the maximum transmission power. The problem of finding transmission opportunities for a given transmission request is then decomposed as:

- Retrieve all rules for the currently active policy.
- Discard the rules which representational hyperspaces do not intersect the hyperspace representing the query.
- For each remaining rule, determine the rule intersection with the query and insert the resulting hyperspace into either one of two lists, depending on the original rule type:
  - the list of "allow" rule intersections
  - the list of "deny" rule intersections
- In an effort to remove potential overlapping sections of hyperspaces from the results, if so required, for each new candidate hyperspace intersection with the query, one can also extract the subset of previously inserted hyperspaces that intersect the candidate to insertion, determine the complements of the candidate once any overlap with existing ones carved out, and insert those complements into the list, in lieu of the whole candidate.
- Apply the disallowed rule intersections to the allowed list, effectively carving out their hyperspaces from the allowed hyperspace list, wherever an overlap is found.
- Depending on the shapes to be carved out, this last step may result in multiple output hyperspaces, for each input allowed hyperspace.

Depending on the expected size of the active policy, in terms of the number of rules, and given the simplification assumption introduced above, whereby geographical areas specified in the rules and the queries are rectangular, one can improve the algorithmic efficiency by indexing hyper-rectangles on each of their dimensions using Interval Trees. For that purpose, our implementation uses an augmented AVL implementation of a balanced binary search tree, that guarantees a worst-case complexity for the needed insertions and intersection lookups as $\mathcal{O}(\log n)$.

Once communication opportunities are determined as stated above, one can simply derive a few useful functions, for example get allowed frequency ranges: given a multi-criteria transmission query that bounds the operational space - time interval and communication parameters - the algorithm returns an ordered list of disjoint frequency ranges that a radio may use to communicate with its peers. The processing follows the steps shown below.

- Evaluate the set of opportunities answering the query.
- Index the opportunities on frequency intervals using an interval tree.
- To ensure resulting intervals are disjoint, for each opportunity candidate for insertion in the tree, split the candidate into complements to readily inserted elements and insert those complements in lieu of the candidate.
- Traverse the tree in-order, while merging adjacent intervals, to create the ordered list of allowed frequency ranges.

We have also implemented a SPARQL-based version of searching for opportunities that is based on a rather complex SPARQL query associated with a couple of registered functions to determine whether hyper-rectangles intersect, and what is their intersection.

6.4.2. Rendezvous Channels

The objective here is to define the Rendezvous Channels Selection as an optimization problem. In other words, the idea is to first formalize the problem and then try to figure out whether this problem is possibly a case of a known problem already formulated (and possibly solved) somewhere else.

We are considering a scenario in which r wireless radio nodes distributed over some geographical 2D space are trying to find a number of channels on which they could communicate, while not violating any of the policies that the radios must obey. For every two radios to communicate at the same time, assuming every pair needs a separate channel (i.e., every pair tuned to the same frequency, while all of the other pairs use different frequencies), we need f frequencies, where f is defined as:

$$f = r(r-1)/2$$

We can think of the value of f as minimal requirement. The Controller will need more frequencies to choose from.

This requirement makes the problem complex, since if there are r radios and f frequencies, there are f^r selection options, of which only f selections satisfy the requirement of tuning to the same frequency. While a full formalization of this problem would require a definition of the tuning precision, conceptually though they must be on *the same frequency* in order to communicate.

The policies are known to all the radios, however, the radios know only their own locations, but not the locations of the other nodes. Moreover, it is assumed that all the nodes must obey the policies, i.e., they can transmit only if there is a policy that allows them to transmit in a given location at a specified time on a given frequency, and there is no disallowing policy. The rendezvous algorithm will be executed on all the radios, with the same input data (i.e., the same set of policies) and thus the result should be the same for all the radios.

We are considering a scenario in which any two wireless radio nodes distributed over some geographical 2D space are trying to find common channels on which they could communicate. The objective here is to define the Rendezvous Channels Selection as an optimization problem. The rules (pertaining to the currently active policy) are known to all the radios, however, the radios know only their own locations, but not the locations of the other nodes. Moreover, it is assumed that all the nodes must obey the rules, i.e., they can transmit only if there is a policy (rule) that allows them to transmit in a given location at a specified time on a given frequency. The rendezvous algorithm will be executed on all the radios with the same input data (i.e., the same set of rules) and is designed to guarantee some number of common channels between any two radios, even though the locations of radios are different.

Problem Formulation

Given:

- A collection of “primary” policies, P (policies that cover all other “allow” policies, geographically; e.g., D, E, F in Figure 6)
- A collection of “secondary” policies S (all ?allow? rules, e.g., A, B, D, E, F in Figure 6)
- A ratio, r, of the number of selections of center frequencies from the policies F_P vs. the frequencies selected from the policies F_S, $r = |F_P|/|F|$, where $F = F_P \cup F_S$
- The number of maximum communication channels, n
- One half of the width of each channel, b
- A set of sub-bands $R = \{R_1, \ldots, R_k\}$ (derived by the opportunities algorithm)
- Allowed frequency multiplier, Δ

Find:

- Center frequencies, $F = \{f_1, \ldots, f_n\}$

Constraints:

- $\forall_{f_i, f_j \in F} \bullet [f_i - b, f_i + b] \cap [f_j - b, f_j + b] = \emptyset$
- $\forall_{f_i \in F} \exists_{R_j \in R} \bullet [f_i - b, f_i + b] \subset R_j$
- $\forall_{f_i \in F} \exists_{m \in \mathcal{N}} \bullet f_i = m \cdot \Delta$
- $|F_P|/n \leq r$

The algorithm depends on the following three functions:

- $\boldsymbol{func_1}$ — builds the list of opportunities with relaxed constraints, considering only the time-frequency map of the geographically all-encompassing rules of the active spectrum access policy, and carving out from there the time-frequency map of the union of disallowed rules, also expanded to the full geographical area covered by the active policy. The resulting set of opportunities is then passed through a “channel provider” filter that converts it to a list of “channels" determined by their center frequencies that obey parametrized constraints and limits the number of such channels returned to $n \cdot r$.
- $\boldsymbol{func_2}$ — builds the list of opportunities with relaxed constraints, considering only the time-frequency map of “allow" rules of the active spectrum access policy, and ignoring the disallowed rules of the said policy. The resulting set of opportunities is also passed through the “channel provider" filter, while limiting the returned set of channels to $n \cdot (1 - r)$
- $\boldsymbol{func_3}$ — consolidates the results by only retaining results from $func_1$ and $func_2$ that are within the fully constrained list of opportunities as computed from the active policy.

The three functions are leveraged in the main control flow of the rendezvous algorithm as follows:

- Retrieve the time-frequency map of the geographically all-encompassing rules of the active spectrum access policy (assuming there are such rules).
- Retrieve the set of “disallowed” rules from the active policy.
- Append the ordered list of frequencies returned by $func_2$ to the list of frequencies returned by $func_1$.
- Then, apply the filter $func_3$ to the list of channels obtained from the previous steps.
- Return the filtered list, ordered, and freed of any duplicates.

6.5. Learner

One of the main responsibilities of the PE is to determine optimal Modulation and Coding Scheme (MCS) values and transmit power (PTx) based on the feedback for the current settings and the history of previously used values for these parameters. The PE automatically invokes this computation on a periodic basis. The proposed transmissions are checked by the Policy Reasoner for the purpose of checking conformance with the active policies. The policy rules are executed by the BaseVISor reasoning engine, which is part of the VIStology Inference Store component.

The conceptual view of the optimization and learning algorithm while selecting proposed values of transmit power and MCS is shown in Figure 8. According to this figure, the PE maintains a representation (a table) of possible transmit states (*knobs*) and relations to other neighboring knobs. Periodically, the algorithm selects a knob and sends it to the radio to use in the next transmissions. The radio then sends feedback to the PE summarizing the performance of the radio using the selected knobs in the current environment parameters. The feedback was computed based on the dependency of BER on Eb/No. After receiving the feedback, the PE updates its state table accordingly. This is the learning step. Then the algorithm selects the next knob to be sent to the radio.

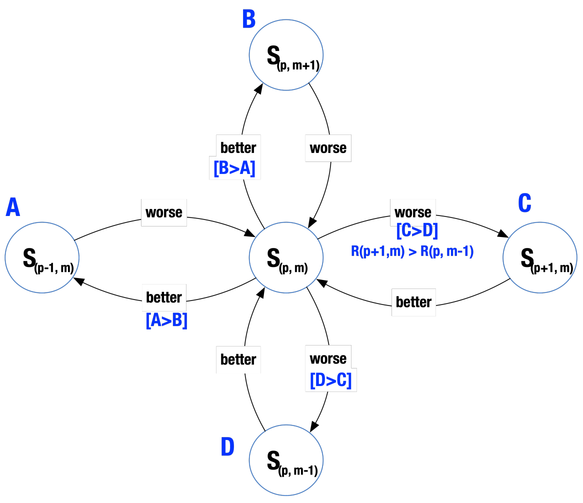

Figure 8. Modification of Transmission Parameters.

In order to analyze the performance of the PE we implemented the initialization and the main loop of the algorithm in Matlab. The conceptual model of the state transitions (Figure 8) was implemented as a table. It is a 5 x 3 matrix in which each row corresponds to a specific value of the transmit power, while each column corresponds to a specific MCS. There are two matrices associated with this model; we call them PTable and KTable. The entries in the PTable represent the expected quality of the transmission using the specific knobs. These values are updated by the learning procedure. The KTable, on the other hand, keeps the probabilities of success weighted by the MCS. The entries of this table were calculated by multiplying the PTable by 2, 3 and 4 for QPSK, 8-PSK and 16-QAM, respectively.

To compute the feedback, we first simulated the Bit Error Rates (BER) of various modulation schemes over an additive white Gaussian noise (AWGN) channel. The input argument for this simulation was, E_b/N_0, i.e., the ratio of bit energy to noise power spectral density, in dB, corresponding to the different E_b/N_0 levels. The feedback calculations were based on these models is shown in Figure 9.

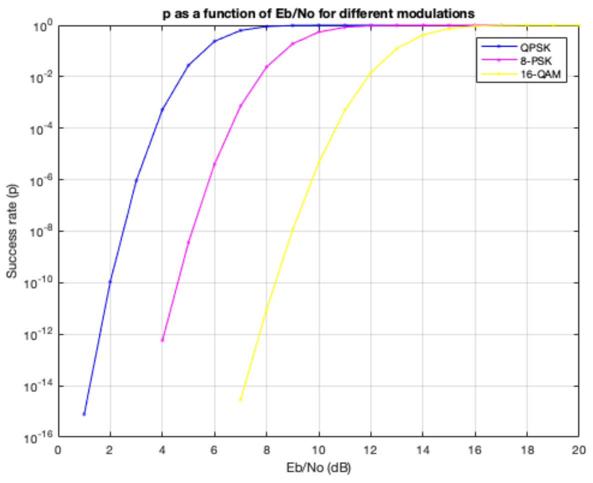

Figure 9. Success rate based on BER models.

We used a simplified reinforcement learning procedure. First, noise was estimated according to the noise model described above. Based on the noise and the power level, P_{tx}, the values of E_b/N_0 was computed and then used to find BER, Packet Error Rate (PER) and p (where N was set to 800):

$$PER = 1 - (1 - BER)^N$$

This was followed by the application of the value of p to the update rule of the $PTable$:

$$PTable(row, col) = p_{cur} + (p_{new} - p_{cur}) \cdot r$$

This was then followed by the update of the KTable, as described earlier and the selection of the maximal value and then using its matrix coordinates to select Power and MCS. So far the above learning rule was used only for a single value update. In the next version of this algorithm we plan to propagate the updates to other neighbors of the current node. The result of simulations are shown in Figure 10.

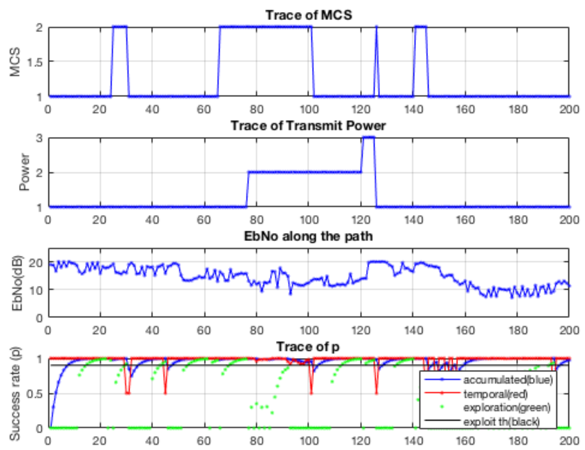

Figure 10. Simulation Results.

As we can see from this figure, in most of the steps the algorithm selects values that guarantee the highest success rate. However, in some steps, the success rate falls very rapidly. Although it recovers in two steps, still this is a rather unexpected behavior, and such a drop is highly undesirable. The explanation of this behavior can be traced to the noise model. The drops coincide with the peaks of noise. There are two issues here. 1. That the modifications of the PTable by the learning algorithm make the table less than optimal. This pattern can be classified as some "instability" of the learning algorithm that results in the selection of less reliable MCS, e.g., 8-PSK instead of QPSK. Additionally, the algorithm does not have means to control the transmitter since the available transmit power is not sufficient.

7. Discussion

As stated in the Section 1, the main objective described in this paper was to experiment with an ontology and policy based system for dynamic spectrum sharing. Towards this objective, we designed and implemented a system whose components were described above. In this section, we ask the question of how can we analyze, and possibly quantify, the impact of not having the functionalities of these modules on the overall performance of spectrum sharing. In short, we are interested in the question of what happens if spectrum management systems are not capable to interpret policies that involve logical rules (higher-level policies). Towards this end, we simulated scenarios in which policies were a mixture of both - hypercube-based and logical rule based policies. The results of these simulations are shown in Figure 11.

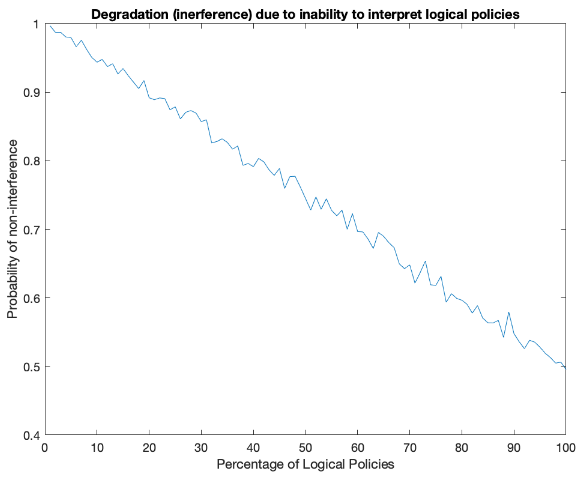

Figure 11. Impact of the missing capabilities of interpreting higher-level policies: simulation results.

This figure shows the dependence of *non-interference* on the percentage of the logical policies as opposed to the hyper-cube based (quantitative) policies. In the simulations, it was assumed that at first the percentage of the logical policies is negligent (on the left) and was increasing to 100% in the end (on the right). The second important assumption was that even if the interpretation is wrong, still, the interference would occur only randomly following the uniform distribution. For this reason, the right-hand side of the plot shows the degradation going down to only 0.5. This assumption is somewhat arbitrary and its validity should be verified experimentally.

8. Conclusions

This paper addresses some issues associated with collaborative spectrum sharing. It describes some of the aspects we identified as relevant and important for any implementation of a spectrum sharing system. The main assumption was that rules for spectrum sharing can change after the deployment of radio networks and the whole system must be able to adapt either on the fly or with as short interruptions as possible. To address such a requirement, we focused on a policy-based approach in which transmissions are controlled by a policy interpreter system, while policies are represented in a formal language (OWL).

One of our primary goals was to develop a prototype of such a system. It is important to notice that three aspects of spectrum sharing, and not just dynamic spectrum access, were addressed. First of all, transmission opportunities were generated based on the active policies. They were all specific to radio locations. This insures that each radio can select transmit according to the local policies. Second, rendezvous opportunities were computed locally, but all of them were based on the set of active policies. Finally, the adaptation mechanisms were implemented in each radio allowing to incorporate spectrum selection based on the propagation conditions at the radios' locations. In summary, this paper outlines a number of issues relevant to dynamic spectrum sharing. However, due to the space limitations the details of the implementation have not been presented.

Acknowledgment

This work was supported by a grant-in-aid of Hanwha Systems and the Agency for Defense Development (ADD) in the Republic of Korea as part of the Contract UC160007ED.

References

1. DYSPAN P1900.1 Working Group. IEEE Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management. Technical report, IEEE, Piscataway, NJ, USA, 2008.
2. Li, S.; Kokar, M.M. *Flexible Adaptation in Cognitive Radios*; Springer: Springer New York Heidelberg Dordrecht London, 2012.

3. R. Studer, V. R. Benjamins, D.F. Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering* **2010**, pp. 161 – 197.
4. W3C. OWL 2 Web Ontology Language Document Overview, 2009. Retrieved from http://www.w3.org/TR/owl2-overview/.
5. J. E. Laird, A.N.; Rosenbloom, P.S. Soar: an architecture for general intelligence. *Journal of Artificial Intelligence* **1987**, *33*, 1–64.
6. Anderson, J.R.; Bothell, D.; M. D. Byrne, S.D.; Lebiere, C.; Qin, Y. An integrated theory of the mind. *Psychological Review* **2004**, *111(4)*, 1036–1060.
7. Endsley, M.R. Bringing cognitive engineering to the information fusion problem: Creating systems that understand situations. Proceedings of Fusion'2011: International Conference on Information Fusion, 2011.
8. Boyd, J. A Discourse on Winning and Losing. Technical report, Maxwell AFB, 1987.
9. Mitola, J.J. Cognitive radio for flexible mobile multimedia communications. Proceedings of IEEE International Workshop on Mobile Multimedia Communications (MoMuC'99), 1999, pp. 3–10.
10. OGC. OGC GeoSPARQL - A Geographic Query Language for RDF Data. Technical report, Open Geospatial Consortium, 2012. Retrieved from http://www.opengis.net/doc/IS/geosparql/1.0.
11. Group, R.W. *Resource Description Format (RDF) Primer*; W3C Recommendation, February 10, 2004. Available at http://www.w3.org/TR/rdf-primer/.
12. Cognitive Radio Ontology. Retrieved on May 30, 2021, from https://cogradio.org/.
13. Moskal, J.; Kokar, M.; Roman, V.; Normoyle, R.; Guseman, R. Towards a SpectralSPARQL Standard for Exchanging EMS Knowledge. Military Communications Conference, 2017.
14. Matheus, C.; Baclawski, K.; Kokar, M.M. BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML and R-Entailment Rules. Proceedings of the 2nd InternationalConference on Rules and Rule Languages for the Semantic Web. ISWC, 2006.
15. W3C. SPARQL Query Language for RDF. W3C Candidate Recommendation, 2006. Available at: http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/.
16. Dumitrescu, A.; Mitchell, J.S.B.; Sharir, M. Binary Space Partitions for Axis-Parallel Segments, Rectangles, and Hyperrectangles. *Discrete & Computational Geometry* **2004**, *31*, 207–227. doi:10.1007/s00454-003-0729-3.
17. Nguyen, V.H.; Widmayer, P. Binary space partitions for sets of hyperrectangles. Algorithms, Concurrency and Knowledge; Kanchanasut, K.; Lévy, J.J., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 1995; pp. 59–72.
18. De Berg, M.; Cheong, O.; Van Kreveld, M.; Overmars, M., Orthogonal Range Searching. In *Computational Geometry: Algorithms and Applications*; Springer Berlin Heidelberg: Berlin, Heidelberg, 2008; pp. 95–120. doi:10.1007/978-3-540-77974-2_5.
19. Guttman, A. R-trees: A Dynamic Index Structure for Spatial Searching. Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data; ACM: New York, NY, USA, 1984; SIGMOD '84, pp. 47–57. doi:10.1145/602259.602266.
20. Beckmann, N.; Kriegel, H.P.; Schneider, R.; Seeger, B. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data; ACM: New York, NY, USA, 1990; SIGMOD '90, pp. 322–331. doi:10.1145/93597.98741.