Policy-Based Cognitive Radios

David Wilkins, Grit Denker, Mark-Oliver Stehr,
Daniel Elenius, Rukman Senanayake, Carolyn Talcott
SRI International,
333 Ravenswood Ave, Menlo Park, California 94025
email: *firstname.lastname*@sri.com

May 24, 2006

To be submitted to *IEEE Wireless Communications, Special Issue on Cognitive Wireless Networks*. Submission due: July 1, 2006, final manuscript due: Dec 15, 2006.
Topic area: Policy Based Cognitive Radio Technologies.

Abstract

We present a new language for expressing policies that allow opportunistic spectrum access while not causing interference. CoRaL has expressive constructs for numerical constraints, supports efficient reasoning, and will be verifiable. The language is extensible so that unanticipated policy types can be encoded. We also describe a Policy Reasoner that reasons about CoRaL policies, and show how this reasoner can be used with various cognitive radios (in this case, an XG radio) to guarantee policy-specified behaviors while allowing spectrum sharing.

1 Introduction

Because of the centralized, static nature of current spectrum allotment policy, wireless communication is confronting two significant problems: spectrum scarcity and deployment delays. DARPA's neXt Generation (XG) Communications Program envisions opportunistic spectrum access [5], which can be realized by achieving the following capabilities:

- Sensing over a wide frequency band and identifying primaries
- Characterizing available opportunities
- Communicating among devices to coordinate the use of identified opportunities
- Expressing and applying interference-limiting policies (among others)
- Enforcing behaviors consistent with applicable policies while using identified opportunities

This paper describes a device-independent policy reasoner that addresses the last two challenges. Our approach ensures radio behavior that is compliant with policies and allows policies to be dynamically changed. The former is achieved by having the policy reasoner either approve or disallow every transmission candidate proposed by a radio, based on compliance with currently active policies. The latter is achieved by expressing policies in a declarative language based on formal logic, and allowing devices to load and change policies at runtime. Instead of inflexibly embedding policies into hardware, firmware, or device-specific software, our approach will enable devices that are flexible and adapt their behavior by changing declarative policies.

This policy-based approach to radio operation decouples policy definition, loading, and enforcement from device-specific implementations and optimizations. One advantage is a reduced certification effort. Accreditation of devices becomes a simpler task if devices can dynamically load policies that govern their behavior. We can certify the policy reasoner and each policy once, independent of the radio, and then test device configurations to see whether they correctly interpret the results from the policy reasoner.

Another advantage of decoupling policies from radio implementation is that devices and policies can evolve independently over time. If a radio does not "understand" a policy, and can thus not fulfill its requirements, it will not have transmissions approved by this policy, thus missing opportunities but avoiding potentially creating interference. On the other hand, if a radio has more capabilities than required by a certain policy, it can just use what is required. Thus, new policies do not require changes in radio software or hardware, and existing policies will work on new radio hardware. Today a cyclic dependency exists where regulatory bodies must wait for technology and technology must wait to see what the policies look like.

The two key components of the policy reasoner are a sufficiently expressive policy language and methods for interpreting policies and supporting efficient policy enforcement with automated reasoning. We have defined the Cognitive Radio (Policy) Language (CoRaL) for use in Phase III of the XG Program (and beyond),[1] and developed a Policy Reasoner (PR) that reasons about policies expressed in CoRaL.

CoRaL must balance the different viewpoints of regulators and radio engineers. Regulators are interested in specifying admissible transmission behavior, but not in how policies are enforced or in how or whether radios can exploit opportunities. Radio engineers, on the other hand, are interested in exploiting as many opportunities as possible, which may depend on how efficiently the PR replies to transmission requests, and how well it communicates possible opportunities when a request is disallowed. Therefore, CoRaL must have a clear, easily understood semantics.

The next section sets the context by describing some assumptions about the PR-radio interface. Subsequent sections describe CoRaL and the PR, respectively. Finally, we discuss related work and draw conclusions.

[1] In Phase II of XG, BBN Technologies developed an XG Policy Language Framework (based on OWL). CoRaL is a new language (based on a typed first-order logic) that is completely different from BBN's language. Some of the ontological concepts for domain knowledge have carried over from the BBN language.

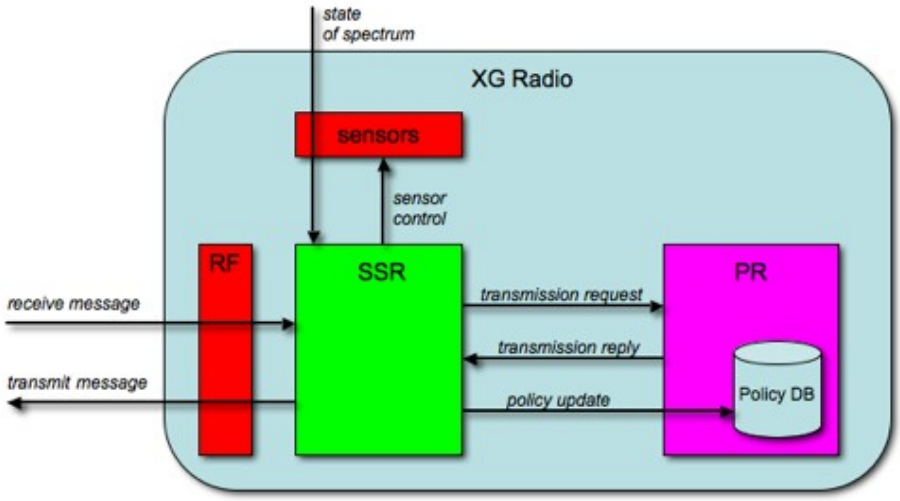

Figure 1: XG architecture. The small boxes are hardware components. The Policy DB is simply a set of policies. The SSR is a radio component that exploits transmission opportunities. This is one of many possible architectures for using our policy language.

2 Reasoning Architecture

At the highest level of abstraction, an XG radio has four main components as shown in Figure 1:

- Sensors. XG radios need sensors to discover available spectrum and transmission opportunities.
- RF. The RF component transmits and receives.
- System Strategy Reasoner (SSR). The SSR controls the radio's transmissions. It builds transmission requests based on sensor data received from the sensors and its current strategies. Replies to its requests from the PR may affect strategy.
- Policy Reasoner (PR). The PR accepts transmission requests from the SSR and checks policy conformance. The PR has all active policies loaded.

It is imperative that the first generation of policy reasoners be easy to verify. Therefore, the PR was designed to be a stateless system. A reasoner that maintained state might be nearly impossible for governing authorities to verify because of the potentially unbounded number of states and the unpredictability of the quality and timing of state updates. Thus, the SSR must assert any required facts about the current state when making a transmission request.

Incoming messages can be control messages, such as updates to policies or system strategies, or messages controlling the coordination with other radios. Similarly, the SSR can create and transmit control messages.

Our focus is the interface between the software components, the SSR and the PR. Before an XG radio can send a transmission, it needs to get approval from the PR. The SSR builds a transmission request, and sends it to the PR. The PR looks at the request and the active policies, and responds by sending one of three replies back to the SSR: (1) The transmission is allowed. The SSR must not transmit unless it has received a message of this type. (2) The transmission is not allowed. (3) The request is underspecified. Given acceptable values of the underspecified request parameters, the transmission will be allowed. The PR returns constraints that must be satisfied.

The SSR can also send policy update messages to the PR, to add or remove policies to/from the PR's set of active policies. The SSR and the PR share knowledge of the CoRaL ontology of domain concepts (such as Frequency, Power), and request parameters. A transmission request generally contains three predefined request parameters, which are assumed to be recognized by every XG-enabled device:

req_radio : Radio; req_radio is the radio requesting to transmit. This parameter contains characteristics of the radio that is attempting to transmit such as detectors on the radio and their thresholds or powermask describing in-band or out-of-band leakage of the radio's transmitters.

req_transmission : Transmission; req_transmission is the transmission that the SSR wants to send. A transmission contains information about the requested band, the requested transmit power, and other parameters.

req_evidence : Pred(Evidence); This is an evidence object that the SSR presents to the PR. It is defined as a predicate over the type *Evidence*, because the SSR might present multiple evidences to the PR. Evidence information can pertain to location, signal, or time or any other evidence that was collected by the radio.

These request parameters are defined in detail in the ontology *request_params*, which can be imported into any policy that defines constraints over transmissions in terms of such request parameters. More details on this ontology are given in the next section. New concepts and request parameters can be defined. We cannot anticipate all future uses, and some users or regulators may have specific needs that should not be part of the common ontology.

Given the open-ended nature of the SSR-PR interaction, how should the SSR behave? What should it put into a request? The XG architecture does not prescribe any answers to these questions. We envision a broad range of XG radio devices, where some may have a sophisticated SSR component, and others may have a simpler SSR.

A simple SSR might put only the requested frequency and maximum power in the request, and hope for approval. If additional constraints are returned, it might just try another frequency from a table, until the request is approved, or the SSR gives up. A cognitive SSR, on the other hand, might exploit spectrum opportunities by constructing a request that satisfies the additional constraints returned by the PR. The SSR might have to perform sensing actions and include evidence for certain requests to succeed.

An example SSR-PR interaction is given in the next section.

3 Policy Language

CoRaL must support permissive policies and restrictive policies. Permissive policies describe conditions under which transmission is allowed, and restrictive policies describe conditions under which transmission is not allowed. For this purpose, CoRaL provides two built-in predicates *allow* and *disallow*, and a policy must have at least one allow or disallow rule. Rules in CoRaL are of the form *allow if formula* or *disallow if formula*, where *formula* describes the conditions under which a transmission request will be granted or denied. Typical spectrum policies restrict or permit transmissions on the basis of frequency, location, time, device capability, node identity, or sensed data. Some simple example policies, encodable in CoRaL, follow.

Frequency band. "Allow transmission between 5180 MHz and 5250 MHz."

Time. "Allow transmission between 06:00 and 13:00 local time."

Location. "Allow transmission if radio is at most 30 miles away from the geographic coordinates (39 10' 30" N, 75 01' 42")."

Node Identity. "Allow transmission if radio belongs to the Red Cross."

Sensed Data - Listen Before Talk. "Allow transmission if radio's peak sensed power is at most -80 dBm and its EIRP for transmission is at most 10 mW."

The design challenge was providing a language rich enough to express numerical constraints (such as frequency ranges, power limits, temporal intervals) and extensible enough to support unanticipated future policies, while at the same time supporting efficient reasoning by the PR. Our solution, CoRaL, is a (subset of) first-order logic with types. Many of the language features were chosen in direct response to the requirements of anticipated spectrum policies. One of the main concepts in CoRaL supporting extensibility and reusability is that of ontologies.

3.1 Ontologies

XG needs a common ontology so that the SSR and the PR, as well as various policy-making bodies and SSR implementers, can consistently and unambiguously refer to radios, radio capabilities and parameters, and the relevant properties of the current radio environment. However, since not all uses of CoRaL can be foreseen, nor all policies or radio capabilities are already known, the language must support user-defined concepts that can be added as needed and used in policy specification.

CoRaL expresses ontologies and domain concepts using type and subtype declarations, and functions and/or predicates defined over types. In fact, ontologies can be specified using the full power of CoRaL, except that ontologies cannot contain an allow or disallow rule. Working with radio experts, we have provided built-in ontologies. For example, we have ontologies for basic types (such as bandwidth, frequency, power), radio capabilities, evidence, signals, time, powermasks, transmissions, and request parameters (among others).

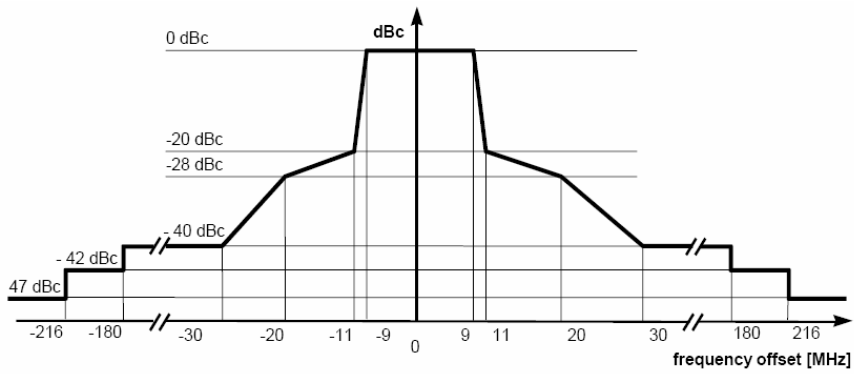

Figure 2: DFS transmit spectral powermask.

The request-parameter ontology in the previous section uses concepts such as *Transmission, Radio*, and *Evidence*. These concepts are modeled in CoRaL as types for which several operations are defined. These complex data structures in turn build on basic spectrum concepts such as *Frequency, Bandwidth*, and *Power*, which are also defined as types in CoRaL (but without operations). An example of an operation on the *Transmission* type is a function that determines the center frequency of the requested transmission:

centerFrequency: Transmission $\longrightarrow$ *Frequency*

Another operation defines the mean EIRP (Effective Isotropic Radiated Power),

meanEIRP: Transmission $\longrightarrow$ *Power*.

The *SignalEvidence* type defines the peak received power:

peakRxPower: SignalEvidence $\longrightarrow$ *Power*.

Another typical ontological concept for spectrum policies is that of a powermask. For example, the Dynamic Frequency Selection (DFS) algorithms for the unlicensed 5 GHz band [3] define the powermask in Figure 2.

Because such powermasks, defined either as graphics or tables, are common to spectrum policies, we support the specification of functions in CoRaL. We provided an intuitive syntax to define powermasks, most of which are either linear or step functions. CoRaL represents a powermask as a list of tuples of numbers *(x,y)*, where *x* refers to the frequency values on the x-axis (in MHz) and *y* refers to the power values on the y-axis (in dBc). CoRaL connects the points either as a linear or step function, as indicated by a keyword preceding the list of tuples. For example, the DFS powermask in Figure 2 is represented in CoRaL as follows:

```
InBandLeakage : Powermask = symmetric
linear [(0, 0), (9, 0), (11, -20), (20, -28), (30, -40)]
step [(30, -40), (180, -42), (216, -42), (inf,-47)] ;
```

The DFS policy requires the average level of the transmitted power not to exceed the limits given in this powermask. Thus, in the DFS policy, we need to compare the

DFS powermask with the powermask of the device. For this purpose, we defined the following functions (among others) in the CoRaL powermask ontology:

```
powermaskLessThan : Powermask,Powermask -> Bool;
powermaskGreaterThan : Powermask,Powermask -> Bool;
```

3.2 Policies

CoRaL policies are formulated over these predefined ontologies, request parameters, and (optionally) user-defined parameters. The following statement uses the basic concepts just described to define our first example policy above ("Allow transmission in the band 5180 MHz to 5250 MHz").

```
policy config1 is
  use request_params;

  allow if
    centerFrequency(req_transmission) in {5180.0 .. 5250.0};
end
```

This policy refers to the request parameter *req_transmission*, which is of type *Transmission*, so the policy can apply the function *centerFrequency* to determine what center frequency was requested. If the center frequency is within the given range, then the transmission will be allowed.

The fifth example policy (a simple Listen Before Talk type) is formalized in CoRaL as follows:

```
allow if
    (exists ?se:SignalEvidence)
      req_evidence(?se) and
      peakRxPower(?se) =< -80.0E-2 and
      meanEIRP(req_transmission) =< 10.0E-3;
```

This policy refers to the request parameters *req_transmission* and *req_evidence*. It states that the SSR request must have at least one evidence object (*se*) with a peak received power of less than -80 dBm. It also constrains the power of the *req_transmission*.

3.3 Requests

A CoRaL transmission request is a set of constraints over the request parameters. For example consider the following request:

```
request R1 is
centerFrequency(req_transmission) = 5500;
inBandLeakage(req_radio) = step[(0,-30),(20,-40),(30,-50)];

se : SignalEvidence;
req\_evidence(se);
peakRxPower(se) =< 30.0;
```

```
end
```

This request gives concrete values for the centerFrequency of the requested transmission, and the inBandLeakage of the requesting radio, but does not specify the power. It also declares a constraint for the peak received power. More specifically, it states that there is evidence being presented (*req_evidence(se)*) that asserts that the peak power received by the radio was less than 30 dBc. Requests cannot include allow/disallow rules or statements that define types or constants.

Depending on the active policies, the PR might respond to this request with *yes*, *no*, or constraints to be satisfied. The latter option will often take the form of a disjunction of possible alternatives. For example, one alternative constraint from the PR may be *meanEIRP(req_transmission)* $<= 20$, which mentions a parameter that is not present in the request. If the SSR understands this parameter, it can produce an allowable request by adding this parameter (with a suitable value) to the request. Another alternative constraint might be that the peak received power must be less than 25, which constrains the value of a parameter already in the request. A simple SSR might ignore this reply and try a different request. A cognitive SSR might (if the radio has the capability) resubmit the request with a lower received power after performing a signal detection that satisfies the policy-requested threshold.

Summary CoRaL has been designed to be extensible and customizable. There are many ways to take advantage of this. Policy authors can specify additional (customized) domain knowledge as CoRaL ontologies, which are imported into their policies and are reusable by other policies. Libraries of customized policies and concepts specific to a particular regime can be developed and used by other authors. Such libraries further raise the level of abstraction as seen by authors.

Further aiding usability, policy authors will not have to see the CoRaL syntax if they use an authoring tool that SRI plans to develop. That tool could be viewed as providing a selection of appropriate abstractions for creating policies. [10] and [9] provide more details on the language and architecture.

4 Reasoning

The core reasoning problem in XG is to infer from a given set of policies and a given set of facts asserted by the radio that a transmission is permitted. While similar to a classical theorem-proving problem, there are several features that distinguish the XG problem.

First, permission to transmit is obtained by an incremental proof, which can consist of several rounds of interaction between the radio and the PR. Depending on how the proof goes, it may require the radio to assert additional facts, which in turn may require additional sensing actions. The evidence provided by the radio would increase monotonically until the interactive proof attempt succeeds or fails.

Second, anytime solutions are important. If the PR is interrupted, the results should be interpretable so that appropriate additional facts can be provided by the radio.

Third, policy authors must be able to predict the behavior of the PR for their policies, which is not usually a requirement for automated theorem provers. Thus, a clear operational semantics is needed and modifications should be avoided. Predictability refers not only to the final result but also to the time required. Thus, equational and logic-programming techniques are preferable to exhaustive search, because their dynamics can be better controlled.

Fourth, a cognitive radio sometimes cannot or does not wish to form a fully specified transmission request. The radio may not be aware of all the applicable policies or may employ a strategy of not initiating costly sensing operations unless required. The PR tells the radio that the underspecified request would be valid if certain constraints were met. Thus, the PR should combine efficient evaluation of fully instantiated requests with reasoning about the more complicated constraints of underspecified requests.

To address these features, our approach is based on a unique combination of functional, equational, and (constraint) logic-programming languages and of automated theorem proving [2, 8]. We reduce the search space of goal-oriented reasoning by using the following techniques. Forward reasoning enables us to quickly prune and filter inapplicable policies, so that they do not impose a burden on subsequent reasoning. Partial-evaluation techniques are used to evaluate terms. A limited form of backward chaining allows us to support user-defined predicates and hierarchies. Conditional equational rewriting allows us to reason with user-defined functions. Constraint propagation and simplification techniques support built-in predicates and functions, and allow us to detect inconsistencies as early as possible.

5 Related Work

CoRaL is uniquely designed for the needs of XG policy reasoning, and enables specification of machine-readable policies that can be efficiently reasoned with. However, it builds on several existing functional, equational, and (constraint) logic-programming languages and on various automated theorem-proving techniques [2, 8].

Other recent work on machine-readable policy and rule languages (e.g., SWRL, RuleML [1], SWSL, Rei [6], and KAoS [4]) focus on a more general functionality. Some of the languages provide tools for analysis of policies, such as checking whether a given situation matches a policy or whether policies are consistent. Rei and KAoS make use of existing reasoning technology: Rei uses Prolog-style reasoning and KAoS uses existing Description-Logic reasoning. Technologies such as SweetDeal and SweetRules [1] provide extensive expressiveness for describing ontologies. Their expressiveness causes them to be less efficient than required for XG. CoRaL is more tightly focused on XG requirements.

Frequency-agile devices require techniques to allow opportunistic spectrum access while not causing interference. We propose to use policies to accomplish this, and an important source of such policies is the Listen-Before-Talk (LBT) policies presented in [7], and the DFS policy already discussed [3].

6 Concluding Remarks

CoRaL is a new language, based on a subset of first-order logic with types, for expressing policies that allow opportunistic spectrum access while not causing interference. CoRaL satisfies the requirements of having expressive constructs for numerical constraints, supporting efficient reasoning, and being verifiable. We have made the language extensible in several ways so that unanticipated policy types can, we hope, still be encoded.

The most important trade-off in the design of CoRaL is the trade-off between expressiveness and tractability. A more expressive logic is able to represent more complicated policies. However, reasoning with them will generally be less efficient. For very expressive languages, the logic will be undecidable – we will not have any guarantees of ever getting back the answer for an inference problem, even given infinite processing power and memory. To achieve an optimal trade-off between expressiveness and tractability, we designed a language tightly around the reasoning required by cognitive-radio policies, avoiding unneeded expressiveness.

Currently, the initial implementation of the PR provides only yes/no answers to transmission requests. We are developing a more capable PR that will reason about the more complicated constraints of underspecified requests, and return constraints that can be satisfied to make a request permissible. We will develop techniques for delegating authority, and activating policies accordingly. Work farther in the future includes a policy-authoring tool, which could be viewed as providing a selection of appropriate abstractions (away from CoRaL syntax) for creating policies, and tools for policy analysis. We also plan to investigate representing policies beyond spectrum-access policies, such as network or security policies.

Acknowledgments.

This research was supported by DARPA's neXt Generation (XG) Communications Program under Contract Number FA8750-05-C-0230.

References

[1] Sumit Bhansali and Benjamin N. Grosof. Extending the SweetDeal approach for e-procurement using SweetRules and RuleML. In *Proc. Rules and Rule Markup Languages for the Semantic Web*, pages 113–129, 2005.

[2] Alan Bundy. A survey of automated deduction. *Lecture Notes in Computer Science*, 1600:153–174, 1999.

[3] Reference DEN/BRAN-002000-2. ETSI Standard EN 301 893 V1.2.2 (2003-06). Technical report, European Telecommunications Standards Institute, 2003.

[4] A. Uszok et al. KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *IEEE Workshop on Policies for Distributed Systems and Networks*, pages 93–98. IEEE, 2003.

[5] XG Working Group. The XG vision. Request for comments. Version 2.0. Technical report, BBN Technologies, 2005.

[6] L. Kagal, T. Finin, and A. Joshi. Declarative policies for describing web service capabilities and constraints. In *W3C Workshop on Constraints and Capabilities for Web Services*, Redwood Shores, CA, 2004. W3C.

[7] Alexe E. Leu, Mark McHenry, and Brian L. Mark. Modeling and analysis of interference in Listen-Before-Talk spectrum access schemes. *International Journal of Network Management*, 16(2):131–147, 2006.

[8] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning*. Elsevier and MIT Press, 2001.

[9] SRI XG Team. XG architecture. Request for comments. Technical report, SRI International, 2006.

[10] SRI XG Team. XG policy language. Request for comments. Technical report, SRI International, 2006.