# Policy-based Network Management for NeXt Generation Spectrum Access Control

Filip Perich
Shared Spectrum Company
Vienna, Virginia, USA
fperich@sharedspectrum.com

***Abstract*—This paper describes the design and implementation of a policy-based spectrum access control framework as part of the DARPA NeXt Generation communications program. The main emphasis of the framework is to overcome two types of concerns: harmful interference caused by a malfunctioning device and harmful interference caused by a malicious user. In tandem with signal-detection-based interference-avoidance algorithms employed by software-defined radios, we design a set of policy-based components, tightly integrated with the accredited kernel on the radio, for avoiding potentially harmful interference caused by a malfunctioning device. The policy conformance and enforcement components ensure that a radio does not violate policies, which define regulatory and other stakeholders' goals and requirements, and which are encoded in an abstract, declarative language. We further secure the policy management and distribution mechanisms in order to prevent malicious users from altering loaded policies as well as from inserting additional policies and thus causing a harmful interference. Additionally, we report on a prototype implementation and demonstration of our framework, which qualitatively illustrates the capability offered to radio for enforcing policies and the capability for managing radios and securing access control to interfaces changing the radio's configuration.**

***Keywords*—Interference Avoidance, Policy, Radio Spectrum Management, Security.**

## I. INTRODUCTION

Smart software-defined radios (SDR) offer tremendous performance and operational benefits. These include the ability to employ dynamic spectrum access methods, to tailor the system design to a user's unique circumstance, and to remotely configure and control networks; however, SDRs pose high security risks. The risks continue to hinder deployment of the technology. For the technology to succeed, SDR developers must carefully examine and mitigate these risks.

The main source of security concerns is the "smartness" or the ability to reconfigure a deployed software-defined radio. It is almost ironic that the main advantage of the technology is also its main weakness. On one hand are concerns focusing on risks due to a radio's malfunction. On the other hand are concerns that focus on security threats caused by a malicious user. Both types severely impact the technology's future.

Some concerns are generic and apply to any technology. These concerns need to be adjusted in order to focus on specific software-defined radio features. For example, although a malicious user could use a software-defined radio to interfere with an existing system, there is other adaptive technology readily available that could cause the same harm. The main concern should instead focus on how much easier it is to use a software-defined radio to achieve the interference versus other available technology.

Other concerns, however, require a careful examination and need to be mitigated as they focus on characteristics and consequences of using adaptive software. Adaptive software can change its operation mode throughout the life of the radio. It is difficult and often unfeasible to verify all states of a software-defined radio. Additionally, a software-defined radio may operate in numerous bands during its lifetime. For each band, the software may need to emulate different processes. As the final processes may be unknown prior to a device's production, the processes may need to change and re-configure dynamically. This raises many issues as to who should perform these operations and what effects these operations can have. This is very different from existing radio technology and significantly impacts current certification practices.

A model is, therefore, necessary that addresses these security threats. Without mitigating the challenges, SDR technology is risking its own existence. If the technology is not accepted by existing commercial and military spectrum stakeholders, it will never be adopted. Moreover, if the technology is not accepted by regulatory stakeholders, it cannot legally be produced and made available to public.

To address the challenges of SDR technology as well as to quantify the benefits it has to offer, the DARPA NeXt Generation (XG) Communications Program [1] is developing novel technologies and system concepts for dynamically redistributing allocated spectrum along with novel technologies to enable spectrum pooling and secondary spectrum sharing. The program focuses on both technical challenges, such as processing power and signal sensing requirements, and acceptance challenges, a subset of which are the security risks.

As part of the XG program, Shared Spectrum Company is developing a policy-based network management framework for controlling spectrum access. This article reports on the prototype implementation and results of the effort that were successfully demonstrated in the summer of 2006. [2]

This work was supported in part by the U.S. Department of Defense Advanced Research Projects Agency XG Program under Contract FA8750-05-C-0150.)

We designed a policy-based approach as policies enable a uniform yet flexible model for controlling devices built by different manufacturers. Additionally, as radios may operate in bands and locations under different jurisdictions and points of control, a policy-based approach allows each controlling body to re-use the model for adjusting devices to suit current requirements as well as for coordinating with other entities.

Our approach consists of three areas. First area defines the foundation for representing policies. It provides the capability to define in a computer-understandable way what a radio can do and what the radio must not do. The area consists of a declarative policy language, types, rules, and instances. Next area defines accredited policy components on the device for enforcing policies by restricting the device's operational state. The last area defines a framework for securely creating, managing, and controlling the device policies.

The benefits of our approach are:

- Ease of configuration – By changing policies and their content, one can modify any configuration, setting, or rule used by heterogeneous devices.
- Ease of policy authoring – By using a declarative language, one can create a generic policy abstracting the low-level requirements of specific devices.
- Secure policy management – Using the management framework one can control what policies a device is using as well as monitor a device for its actions. The framework is further secured for limiting who can control the devices.
- Secure policy distribution – Using the policy distribution framework, one can securely transmit policy commands and queries to devices as well as receive responses.
- Secure policy enforcement – Policy components are part of an accredited kernel, which has a direct control over the transmitter and can thus impose limits on transmitted power and frequencies.
- Automated policy synchronization – Policy components automatically identify stale policies and either request an update or stop the device from transmitting until the policies are updated.
- Automated policy conflict resolution – Policy enforcement component automatically combines all policies and permits transmissions only if requirements imposed by all applicable policies are met.
- Automated opportunity discovery – Policy components automatically determine sufficient device states that would satisfy policy enforcement requirements and make them available to the device as additional spectrum opportunities.

In the present paper, we describe the policy-based framework and explore the benefits it provides. In Section 2, we describe the overall policy-based control architecture, the language, and the processes. In Section 3, we offer details on implementation and prototype demonstration. In conclusion we provide suggestions for future work.

## II. Policy System Architecture

### A. Regulatory/ Control and Configuration Framework

Smart software-defined radios and their security challenges have recently attracted vast interest and spawned topics of discussions across many communities. However, many of the questions raised are familiar; many of them were raised with other disruptive technologies. One thing that sets SDR apart from some of these technologies is the sheer scope of their potential, and as a result, the variety of interested parties they affect. A unique aspect of software-defined radios is that they propose a radically different approach from currently established principles that affects many stakeholders.

The stakeholders – represented as actors – and their impact on software-defined radios are illustrated in Fig. 1. Although, the actors are the same as for any other wireless technology, the difference is where each actor can influence the design, implementation, and deployment of a radio.

XG users and XG operators range from commercial and military bodies that are interested in pooling spectra together in order to improve their current spectrum access. There are also other XG users and XG operators who are interested in leasing spectrum or using spectrum available to the second market. In order to differentiate current users and operators using the spectrum from potential users and operators of XG software-defined radios, without loss of generality we refer to the latter as XG users and XG operators.

Next there are spectrum owners, users, and regulators who are excited about sharing spectrum with others or making sharing more available. At the same time, these "actors" require an assurance that the software-defined radios will not interfere with current spectrum users.

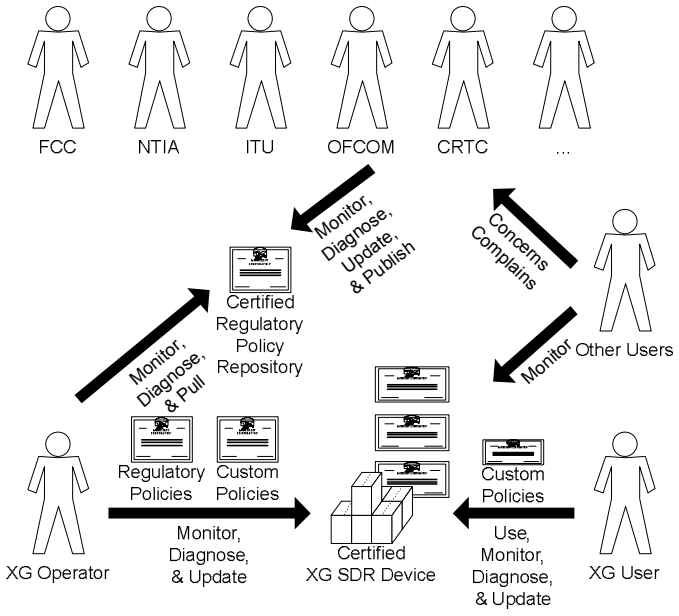


Figure 1. A policy-controlled software-defined radio needs to enforce requirements of operator, user, regulatory bodies and other spectrum stakeholders.

The traditional model employs a *prevention mode* to eliminate security threats. First, regulators define fixed requirements for building an application-specific radio. A manufacturer designs, produces, and proves to regulators that the built radio satisfies the requirements in order for the radio to be classified as "certified". Only then can an operator or user purchase such a radio. Although the radio device may be configurable, the configuration options are minimal in comparison to the capability of configuring software-defined radios, and thus for the lack of better words, the radio is "certified and locked". Therefore, devices are open to modification during manufacturer's design and production phases only. In subsequent steps, devices are already locked. This implies that a manufacturer selects and initially configures devices. There is no support for future updates to manufacturer or regulatory policies on the device. Additionally, there is only a limited, indirect support for future updates to an operator's custom policies affecting the device. These updates are external to the device. There is no support for explicitly enforcing regulatory, manufacturer's, or operator's policies on the device. Security threats are thus mitigated – prevented – by design.

For SDR design we advocate an *avoidance mode* for eliminating security threats. Regulators still define fixed requirements; however, the requirements are now for a generic radio. A manufacturer designs, produces, and proves to regulators that her radio satisfies the requirements in order for the radio to be classified as certified. The device, however, contains software modules that allow for the radio to be re-configured dynamically at later times. Therefore, the radio is "certified and open" as it can operate in different bands emulating different applications. At the same time regulators also define regulatory policies, which are applicable to specific applications and which are certified by the regulators for use. In this operating mode, the XG operator purchases an open, certified device from a manufacturer. The XG operator then specifies the type of application the radio is to be used for by providing the device through a secure link with appropriate certified policies. The SDR device now consists of two certified objects – (i) a policy component, as part of the certified radio, which ensures that the radio conforms to policies, and (ii) the certified policies, which clearly define what the radio can do and what the radio must not do. Therefore, security threats caused by malicious users are prevented through enforced security guards on the device and on the interface links as any other device would do; however, security threats caused by malfunctioning or modifying a device are now eliminated by dynamically avoiding them.

This approach addresses concerns raised by any stakeholder that is affected by smart software-defined radios. The approach allows regulators and other points of control to continue controlling where and how software-defined devices are allowed to transmit, yet at the same time this approach allows XG operators and XG users take the full advantage of a software-defined radio technology.

### *B. Policy Architecture Components*

While the policy-based architecture advantages are clear, the complexity lies in the details. How can a software-defined radio be securely modified, directly and indirectly, by users, operators and regulators? How can a software-defined radio change its functionality by simply uploading different policies? How can a software-defined radio decide what policies to enforce? How can a software-defined radio resolve policy conflicts?

To address these questions, we divide the policy architecture intro three areas. The overall architecture is depicted in Fig. 2.

The first area represents a policy language with a set of policy classes and their respective instances. We delve into the details in the next section.

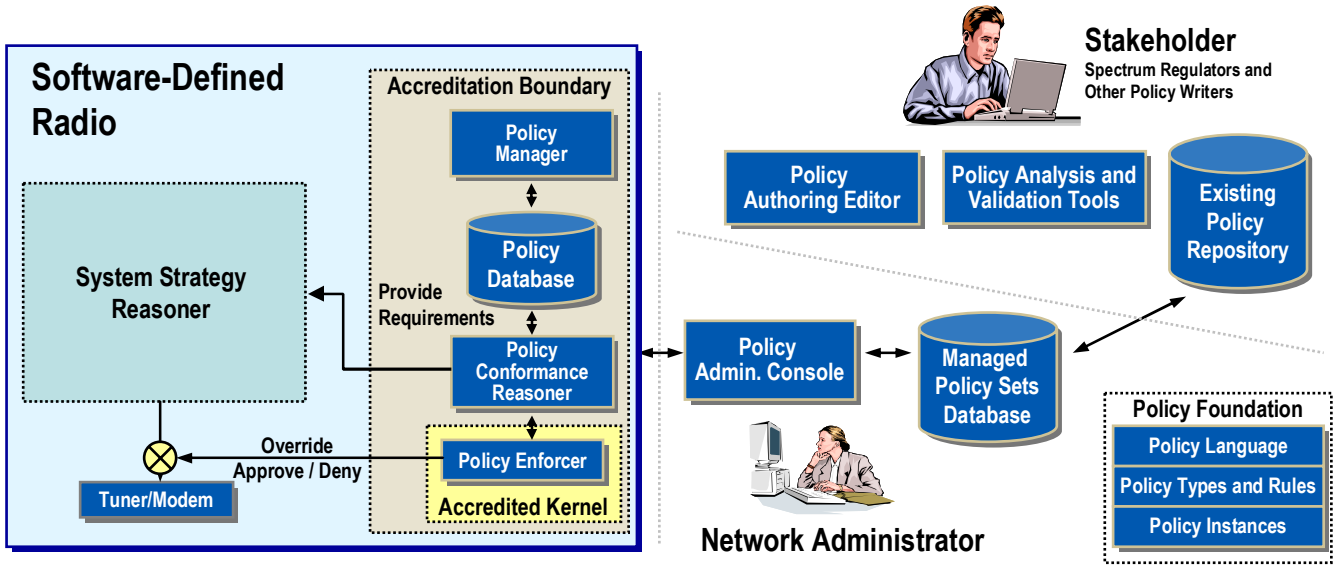


Figure 2. Distributed Policy-based Spectrum Access Control Architecture. The software-defined radio includes a localized policy enforcer responsible for controlling spectrum access based on requirements provided by regulatory and other stakeholders. The radio is managed by a network administrator who relies on stakeholders for providing up-to-date policies.

The second area represents on-node policy components. These are components that are present on every software-defined radio platform, although as is illustrated later in this article, some of the components may be transferred onto a proxy controller. The components include a policy manager for managing local policies and responding to remote commands, a policy conformance reasoner for reasoning over policies and device-provided evidence, a system strategy reasoner for adjusting and selecting the device's operational mode, and a policy enforcer for governing the radio by permitting only allowed transmission requests.

From the policy perspective, the system strategy reasoner (SSR) represents an external module that controls the hardware, gathers data, forms strategies, and acts as an interface for transmitting and receiving data.

The SSR is responsible for interacting with the policy conformance reasoner in order to determine what spectrum opportunities are currently available. Using that information, the SSR can determine and execute applicable system strategies that are needed in order to for the radio's transmissions to conform to policies.

Additionally, the SSR is responsible for providing the policy component with an access to its current state as well as evidential results obtained from other radio components, including detectors, and data received from other radios.

It is, however, the policy enforcer (PE), which is responsible for assuring that the device does not cause harmful interference by enforcing that the SSR configures the radio in one of the approved opportunity states and by filtering transmission requests sent to the transmitter on the software-defined radio.

The last area represents off-node policy components used by policy administrators and analyzers. For analyzers, this area includes policy authoring editor together with policy analysis and validation tools. For administrators, the area consists of a manager console and a policy dissemination framework.

The Policy Administration Console provides a secure interactive method for operators to remotely modify the state of one or many radios. The operators are able to modify both policies and operational modes in which a radio or group of radios operates. The console relies on a X.509 Public Key Infrastructure (PKI) for authorization, authentication, and accounting [3]. Additionally, the console relies on a secure dissemination framework for securing direct and multicast links between the console and devices. Through the secure link, the console communicates with a Policy Manager located on the smart radio or on its proxy.

The details of policy management and provisioning are described in the subsequent section. Before we describe how policies are uploaded onto a device and how they are employed by the device, we must first define the policies.

### C. Policy Language

The XG Policy Language represents a declarative, semantic language for expressing policies and logic used for guiding operation of software-defined radios. Its main purpose of the language is to allow regulators, operators, and users to define abstractly requirements for controlling access to a spectrum, using a policy-driven evidential approach.

The language defines concepts for expressing knowledge about a radio frequency device, its components, capabilities, and current state. The language also defines concepts for expressing restrictions on the devices and its components. Finally, the language defines concepts for combining the restrictions and definitions into rules and policies.

The language is based on the W3C Web Ontology Language (OWL) [4], which provides an interoperable, machine independent language for expressing information that can be processed by humans and by software applications. Using OWL, one can create ontologies, which define vocabularies for representing meaning of a subset of domain-dependent terms and the relationships between those terms. Using these ontologies, one can annotate information that can be shared and used to infer additional information across heterogeneous domains, applications, and platforms.

Additionally, the language employs concepts from Deontic logic for expressing actions a device can undertake. The language focuses on two concepts – permissions and prohibitions. Permission defines what a device is allowed to do and the constraints the device must first satisfy. Prohibition, on the other hand defines a situation when a device is forbidden from taking a specific action.

The fundamental modeling primitive of the language is *xg:Policy*. This construct is used to represent any policy expressed in the language. A policy defines a collection of facts and constraints that can be used for deciding whether a policy is applicable to current state of a device, i.e. a policy controls the device's access to spectrum.

A permissive policy, *xg:PermissivePolicy*, represents a special policy subclass that permits devices to access a spectrum whenever a device and its evidence can satisfy the policy's constraints.

A prohibitive policy, *xg:ProhibitivePolicy*, is another policy subclass. It prohibits devices to access a spectrum whenever a device and its evidence violate one of the policy's constraints.

The relationships of the three terms are shown in Fig. 3.

Software-defined radio may operate over multiple concurrently applicable policies. For example, a radio may have policies from two regulators, each regulating one spectrum band the radio is capable of operating at. Moreover, some policies may overlap for the same frequency band where the device is attempting to transmit.

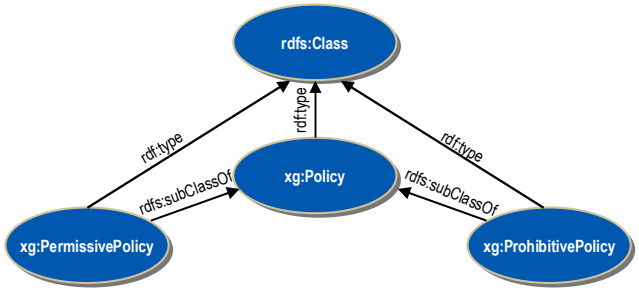


Figure 3. XG Policy Language model.

In order to support multiple policies, the language defines vocabulary for creating meta-level policies. Meta-level policies are used for guiding the operation of access control policies, i.e. for guiding permissive and prohibitive policies. The meta-level vocabulary defines constructs for de-conflicting overlapping policies.

A default de-confliction rule of the language states that a prohibitive policy overrides permissive policy. At the same time, the meta-level vocabulary allows one to define absolute and relative prioritization of policies, thus overriding the default rule. The language defines vocabulary for assigning numeric priority levels to policies. The language also defines vocabulary for relatively ordering policies by defining relationships between pairs of policies.

In addition to policies, the language defines vocabulary for expressing knowledge about the state and capabilities of a device. Using this vocabulary, one can define, express, and inter additional information about different devices and their capabilities. Additionally, the vocabularies include terms for expressing state of each functional aspect of a device. This may, for example, include an operational configuration of a transmitter in terms of power and frequency as well as history of collected detections for a signal detector, the latter representing an *evidence*. The language, however, does not define constructs for expressing how the knowledge was contained.

In order to express the evidence and states of software-defined radios, the language defines domain vocabulary for custom networking and electrical engineering concepts. This includes, for example, vocabularies for expressing and reasoning about time, location, frequency, power, and signals.

The language also defines vocabulary for expressing conditions on the device's state. The language uses the W3C Semantic Web Rule Language (SWRL) [5]. SWRL, in turn, builds upon OWL abstract syntax and defines vocabulary for representing Horn-like rules.

The language thus allows one to define a policy for controlling spectrum access by specifying desired radio states and collected supporting evidence and by restricting undesired situations.

As shown in Fig. 4, a policy consists of three sections. First is a meta-definition for allowing a policy component to determine when the policy is applicable and how it affects the radio. Next is a set of information the policy depends on. Finally, the last section consists of rules for determining when a device either meets or violates the policy's requirements.

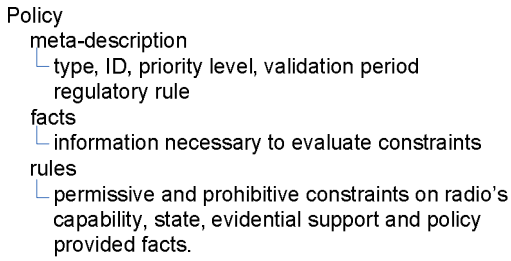


Figure 4. A policy consists of meta-description, facts, and constraint rules.

### *D. Policy Management and Provisioning*

Authenticated and authorized stakeholders create and modify policy documents using the Policy Authoring Tool. Additionally, the stakeholders employ Policy Analysis and Validation Tools in order to analyze the effects and validate the correctness of the policies. This process mostly involves verification that the resulting policy represented in the XG Policy Language correctly represents the intended meaning of original policies, which are often described in plain English and thus subject to interpretation.

Using a secure communication channel, the stakeholders make the policies available to XG operators and XG users. Depending on accountability, either XG operators or XG users are responsible for obtaining policy updates as they become available. This role will be, however, mostly applicable to XG operators only.

The XG operator is responsible for pushing policy updates into appropriate devices. If the device does not receive proper policy updates, the operator is risking that the device will lock itself and will need to be reset, re-loaded with new policies, before it can be used again.

The XG operator employs the Policy Administration Console for communicating with its devices. Each operator is assigned at least one X.509 certificate and a matching private key. Similarly, each radio is assigned another X.509 certificate and a matching private key.

When operator wishes to send a message to a radio, its Policy Administration Console creates a specific RPC command according to the IETF NETCONF [6] format, an XML-based [7] replacement for the IEEE Simple Network Management Protocol (SNMP) [8].

The message format is based on XML. Each RPC command or reply element is an XML element defining one of the specific commands or responses supported by the Policy Manager located on a software-defined radio or its proxy. The element includes a unique message identifier, timestamp, sender identity, and indented recipient identity. For commands, the element also includes the method specifying the type and the content of a command. For feedback responses, the element includes a unique message reply identifier, an error or an ok element, and an optional data element populated with requested content.

Additionally, the RPC element contains an XML digital signature for validating the authenticity and authorization of the RPC command and its issuer. The signature contains a digest hash value of the RPC message, the key information from X.509 certificate, and a signature value signing the entire signature element using the issuer's private key.

Once constructed and signed, the Policy Administration Console sends the message over a secured link to the destination where the message is recognized as either a policy command or query and sent to the local Policy Manager for processing.

Policy Manager acts as a gateway to the accredited policy components located on a device. The on-device policy component architecture is illustrated in Fig. 5.

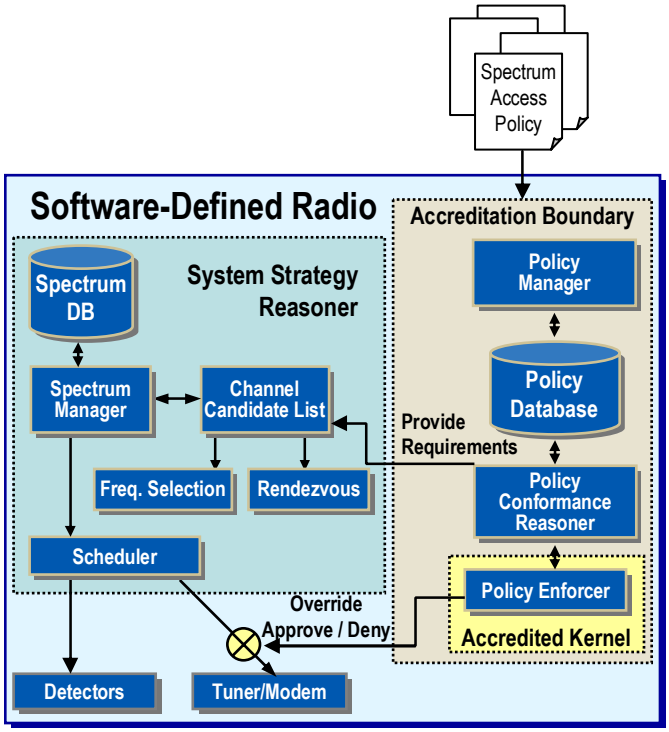


Figure 5. Policy components of an autonomous software-defined radio.

As an alternative, some of the components may be located on a proxy controller. This is particular important for devices with limited computational resources or for client devices in an infrastructure-supported server-client networks. The modified architecture is illustrated in Fig. 6.

The Policy Manager first checks the message for security by validating authentication and authorization of command issuers and sources. The identity of the issuer, the sender, and the destination must be included in the signed message. Using this approach, Policy Manager verifies the integrity of a command, the authenticity of the sender, the authorization of the sender to issue such command, authenticity of the receiver; and the timeliness of a command for avoiding replay attacks.

If the Policy Manager is able to verify that the message should be processed, the Manager checks the type of the message in order to either adjust the policy component state or to answer specific policy-related inquiry.

The Policy Manager is responsible for providing a persistent storage for received policies and for loading active policies into Policy Conformance Reasoner. Policy Manager supports multiple policy configuration modes. Each configuration mode represents a set of policies that are applicable when the mode is "activated". Exactly one mode can be activated at a time and that mode is tagged as "running". All policies that are part of the running policy configuration mode are automatically loaded and activated in the Policy Conformance Reasoner.

The Policy Manager allows operators to add and remove policies from any configuration mode. When a configuration mode is not specified, the current running mode is assumed by default. The Policy Manager also allows operators to switch between modes. When a mode changes, all policies belonging to the previous mode are unloaded and all policies from the new mode are loaded and activated. Hence, by switching a mode, policy administrators can quickly switch between a set of policies.

The Policy Manager also allows operators to query state of each configuration mode. Unless declared otherwise, the current running mode is assumed by default.

Additionally, the Policy Manager enables operators to query state of each configuration mode. Each configuration mode consists of a set of policies. Additionally, the running configuration mode is associated with a set of decisions, complaints, conflicts, and the overall status of the Manager to allow operators to monitor the health of their systems.

The received message may contain a command that affects the current configuration mode. Either a policy may have been added or removed from the mode or another mode may have been activated. In these situations, Policy Manager is responsible for activating the right set of policies inside the Policy Conformance Reasoner (PCR).

From a typical policy-based network management perspective, the PCR functions as a local policy decision unit.

The PCR is responsible for parsing and validating a policy. It checks that the policy conforms to XG Policy Language definitions. The PCR also verifies the validity of the policy by examining the policy's meta-description.

Once validated, the PCR converts the policy into its internal representation. The PCR extracts the data defined inside the policy document as well as extracts and optimizes the constraint rules defined by the policies.

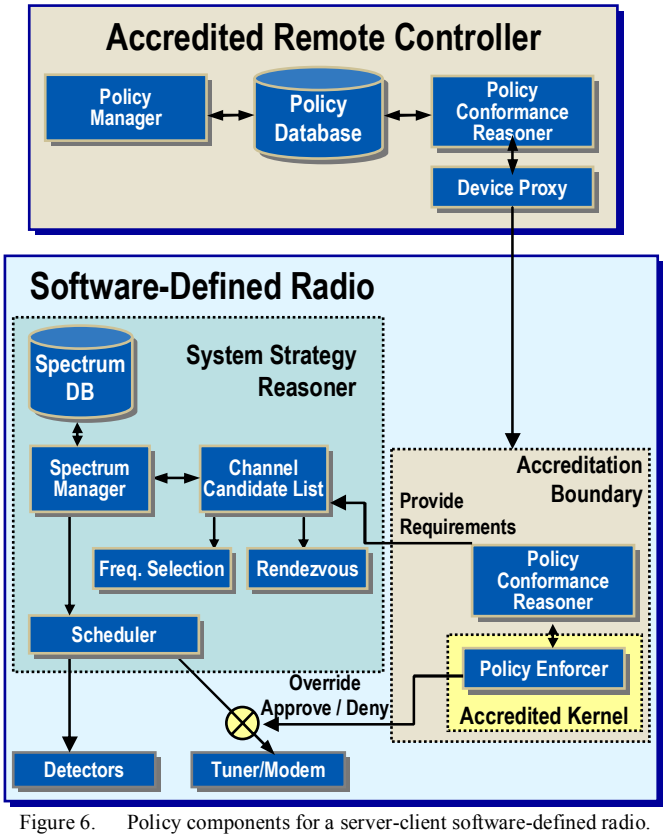


Figure 6. Policy components for a server-client software-defined radio.

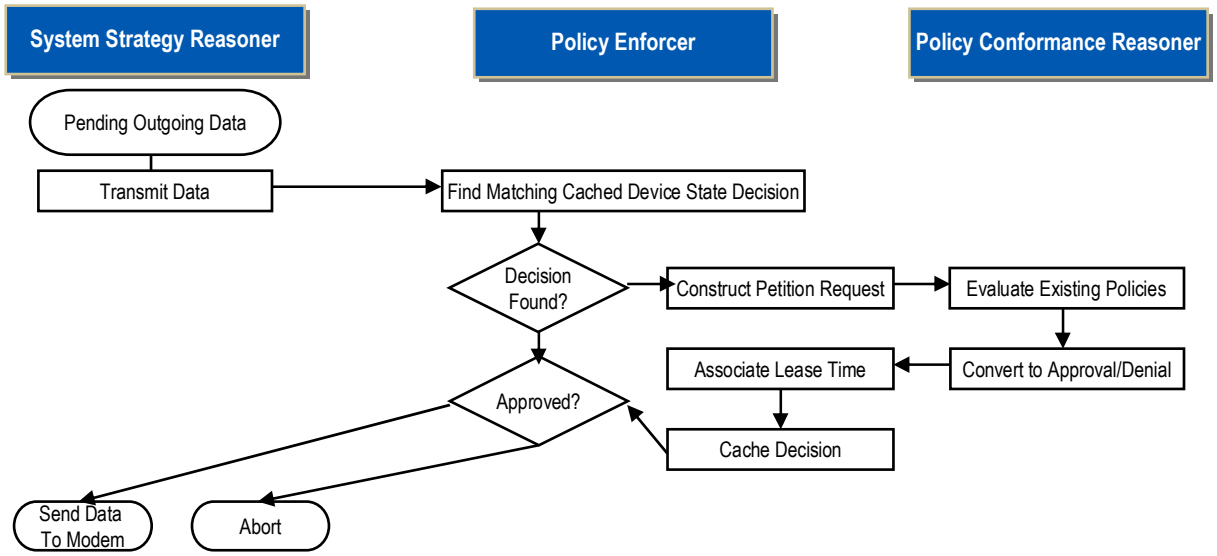


Figure 7. Overview of spectrum access enforcement.

Additionally, the PCR uses the policy's meta-description in order to insert the policy in the sorted list of all active policies based on decreasing importance. This is necessary for reducing the workload required for the PCR to reach a decision on approving or denying a specific transmission request as well as for computing available spectrum access opportunities.

Therefore, the PCR dynamically merges and deconflicts policies as they are made available to the radio. The de-confliction is based applying the default rule for breaking ties between permissive and prohibitive policies, numerical priority levels assigned to policies, and relative policy ordering. Even in the event of one permissive policy and one permissive policy being more important than each other, i.e. creating a cycle, the default rule guarantees that a prohibitive policy takes precedence and thus avoids potentially harmful interference by rather denying instead of allowing a request.

The PCR computes decisions allowing or denying transmissions based on requests originated from the Policy Enforcer (PE).

The PE periodically evaluates the current state of the device and the logical channels employed by the System Strategy Reasoner. For each channel, the PE maintains a set of pre-approved device states that the SSR must match in order to be permitted to transmit. Alternatively, for each channel the PE inquires with the PCR if the current state for that channel would be approved. The PE maintains these decision caches as to limit the amount of work required from the computation-intensive PCR.

Ultimately, the PE pro-actively monitors channels the SSR is attempting to use and enforces that transmissions originating at the SSR fully satisfy policy requirements.

### *E. Spectrum Access Enforcement*

The primary function of the Policy Enforcer (PE) is to avoid harmful interference by interrupting transmission commands sent to a modem on a software-defined radio.

For that the Policy Enforcer maintains a set of pre-approved state models based on configuration policies and associated validity time period. During the adjustable time period, the Policy Enforcer assumes that the pre-approved device state would in fact be approved. The time period may range from zero to potentially hours as defined in, for example, the Dynamic Frequency Selection (DFS) standard [9].

The detailed logical flow of sending data to a modem while enforcing policies is shown in Fig. 7.

When the System Strategy Reasoner (SSR) attempts to send a transmit command to the local modem, the command is interrupted by PE.

The PE attempts to find a matching cached device state decision. The PE evaluates the current radio state, which includes current configuration, capabilities, and results from detectors against pre-approved states. A device state tree is shown in Fig. 8. If a matching decision is found, it proceeds based on the approving value of the decision. If the cache shows that a previous matching request was approved, it is also approved, otherwise it is denied.

Alternatively, the PE constructs a full Petition request and associates it with the current snapshot of the device. The current snapshot represents the capabilities, configuration states, and evidence for each component present on the device. As some data is fairly static, this information is pre-loaded inside PCR upon start-up or component change, and does not need to be provided with each Petition request.

PE sends the request to the PCR, which evaluates it against the ordered list of policies based on decreasing performance. For each policy, the PCR checks the requirement constraints in order to decide whether the policy is applicable to the petition. If the policy is applicable and if its rules are met, the PCR uses that policy as the decisive one.

Each rule associated with the policy represents a set of logical and computational constraints on any capability, configuration state or evidence provided by the device. We have identified 18 policy rule categories based on the types of constraints used in official regulatory policies. The categories

TABLE 1. POLICY RULE CATEGORIES

| |
|---|
| Listen-before-Talk based types |
| LBT – Same up and downlink frequencies |
| LBT – Different, but known, up and downlink frequencies |
| LBT – Different, but known, up and downlink frequencies, and plans |
| LBT – TV band |
| Spatial types |
| Geographic border field strength limits |
| Database geographic/TV coverage area based |
| Temporal types |
| Time of Day restrictions |
| Finite time duration authorizations |
| Device based types |
| Ability to measure second and third harmonic |
| TX power spectrum density limit |
| Adjustable S/N Limits |
| Connectivity based types |
| Beacon reception |
| Connectivity requirements |
| Group based types |
| Bell-ringer group behavior |
| Group-based max power estimation |
| Node identify restrictions |
| Distributed control based types |
| Automated policy updates |
| Automated policy revocations |

are listed in Table 1. For each category, the rule may consider up to several dozens of parameters from Fig. 8.

Therefore, if this policy was permissive, then the petition is approved. Alternatively, if the policy was prohibitive and its rule was satisfied, then the petition is denied.

At this point, the PCR can stop the evaluation. Since the policy list is ordered by a decreasing priority, no policy of a higher priority can override the reached decision.

The decision is returned to the PE, which caches it for a pre-defined time period and which either blocks or allows the transmission to proceed based on the decision value.

### *F. Spectrum Access Opportunity Discovery*

In addition to permitting spectrum access, the policies can be used to determine spectrum access opportunities. This allows the System Strategy Reasoner (SSR) to recognize automatically newly available channels and requirements that the SSR must meet prior to transmitting on those channels.

The logical flow of this process is outlined in Fig. 9.

The SSR first prepares an opportunity selection request, which is similar to a transmission petition request, except that it does not fully populate all parameters of its state. For example, the SSR may choose not to set a transmission frequency and only provide the possible frequency ranges it can or wants to support as a capability. The SSR submits the request to the PCR.

The PCR evaluates the request against locally available policies in order to discover missing values of unpopulated parameters that would render the request a valid. In other words, the PCR populates missing parameters, if possible, that

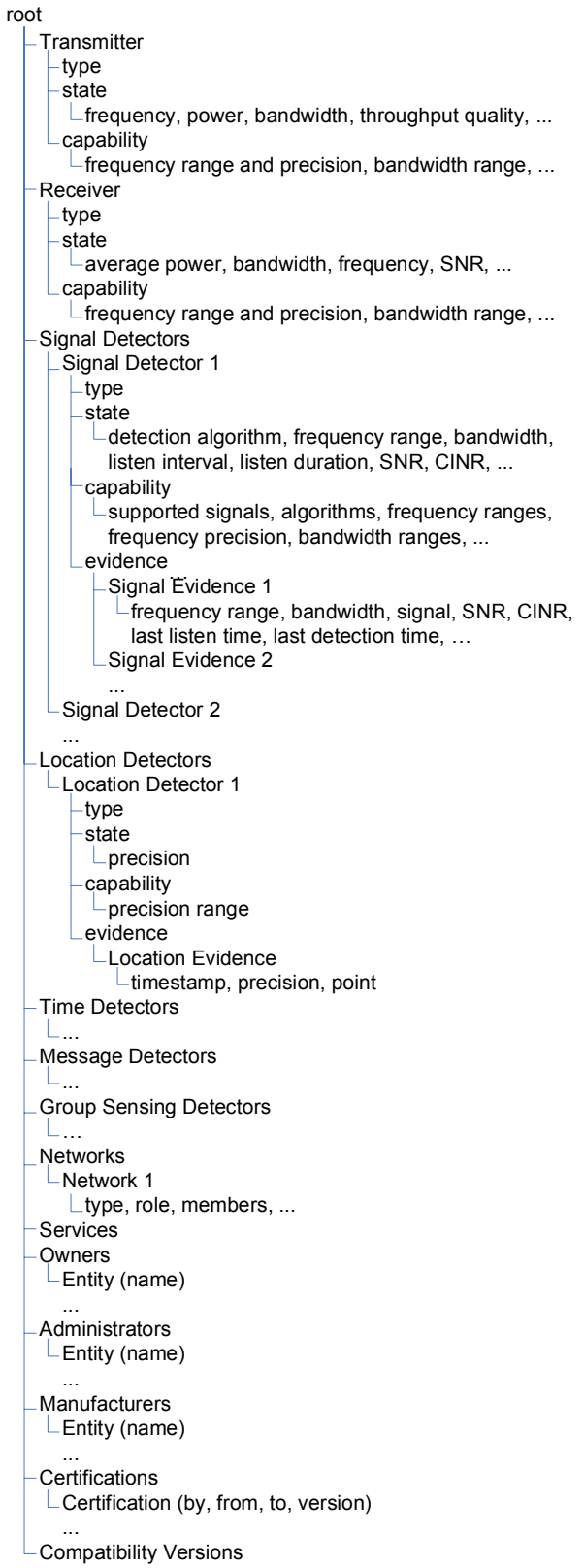


Figure 8. Information maintained about device and its current state.

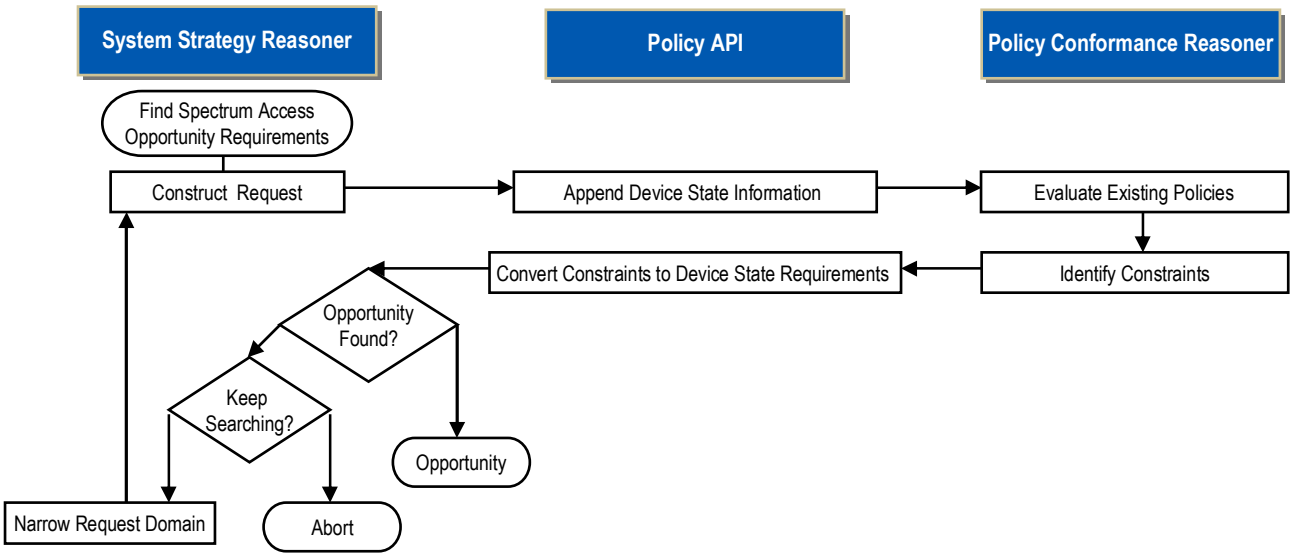


Figure 9. Workflow for discovering available spectrum access opportunities.

are necessary for approving the device state for a transmission. For example, the PCR may conclude that in order for the submitted configuration to be approved, the transmission frequency must be either 5180 MHz or 2310 MHz. Therefore, in this case, the PCR returns a list of two opportunities.

If found, opportunities are translated by a proxy the Policy API component into two device states one representing a configuration with a transmission frequency set to 5180 MHz and another one with a transmission frequency set to 2310 MHz. The two device states are then returned to the SSR.

By evaluating the two opportunities, the SSR can conclude that it needs to monitor for non-cooperative signals at either frequency in order to be allowed to transmit at those frequencies, and adjust its detector configuration and workflow accordingly.

There are, however, situations when the PCR will be unable to find or fully populate an opportunity. If the request does not match any policy or if it violates a policy, then no opportunity is found. Additionally, a parameter may not be bound to a value if there is an unlimited set of possible values. For example, while a value may be restricted to a certain range, depending on the accuracy it may be very expensive to bound a device's position to be within continental USA. Therefore, while for most cases the PCR is able to find a spectrum opportunity; it may be unable to do so in every situation.

## III. System Demonstration

### A. Prototype Implementation

The policy prototype implements the Policy Administrator Console, Dissemination Framework, Policy Manager, Policy Database, Policy Conformance Reasoner, and Policy Enforcer. The prototype also implements a Radio Console for demonstrating and testing the on-board policy components. Additionally, the on-board policy components are integrated with the System Strategy Reasoner and other components demonstrating full capability of an XG-enabled software-defined radio.

The Policy Administration Console is developed as a Microsoft Windows application. The recommended hardware requirements are equivalent to those offered by an off-the-shelf desktop computer.

Fig. 10 shows a sample snapshot of the console. Using the console, an authorized operator can query status of any radio in its domain. The operator can choose between communicating with one or with a set of radios at a time. Through the interface, an operator can send commands to remote Policy Managers ranging from adding and removing policies, activating policy modes, and deleting log entries. Additionally, an operator can use the interface to obtain a list of currently active policies on a certain radio as well as preview historic decisions made by the radio.

The Policy Manager, Database, Conformance Reasoner, and Enforcer are developed using C/C++. For the demonstration purposes reported in this article, the Policy Conformance Reasoner employs a Prolog provided by SWI-Prolog [10]. As the policy components are embedded in a resource-limited device, the SWI-Prolog is replaced with a simplified reasoner also developed in C/C++. While the hardware requirements are designed for an embedded system, the demonstration prototype described in this article is part of the Test Console, which is also a Windows-based application loading the policy components as a dynamic library – DLL. The prototype relies on OpenSSL [11] for a security library. Additionally, for a persistent database, the prototype relies on SQLite [12].

Fig. 11 shows a sample screen snapshot of the Test Console. The console provides interface for modifying the radio's state and thus emulates the inputs from the System Strategy Reasoner. Additionally, the console provides an interface for recording complaints and conflicts in order to emulate additional inputs into the policy components. Most importantly, the console provides interface for accessing the most important functionality of the policy components – obtaining transmission permissions and discovering spectrum access opportunities.

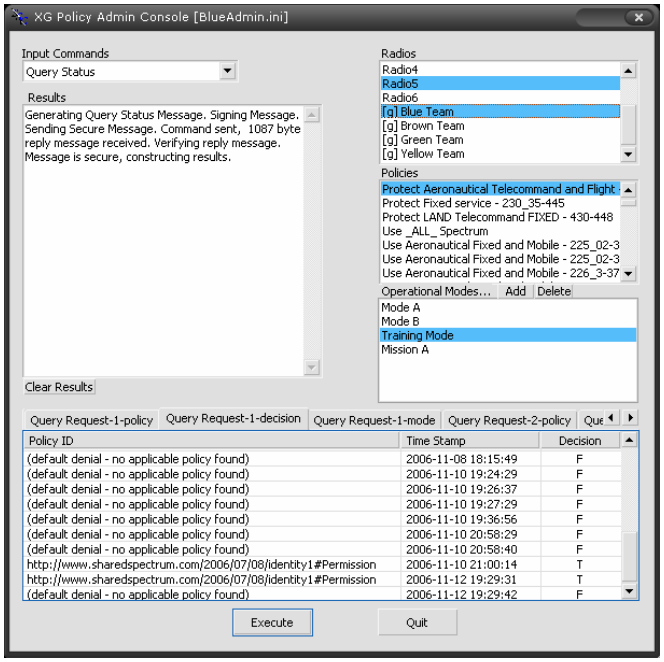


Figure 10. A sample screen snapshot of a Policy Administration Console for demonstrating the capability of policy-based management.

### B. *Prototype Demonstration*

The XG Policy components were officially demonstrated to military and commercial organizations in summer 2006 as part of an overall XG capability demonstration.

The focus of the policy demonstration was to illustrate the capabilities of Policy Manager, Conformance Reasoner, Enforcer, and Administration Console.

We have designed a scenario consisting of two independent administrators and a set of XG-enabled devices, represented by Test Consoles. Each administrator was in charge of its "color" team. Additionally, each radio was assigned a membership in two or more teams. Consequently, each device was managed by one or two operators. This illustrates a scenario when a device operates in two regions with different points of control or in one region with multiple points of control. Regions can be divided geographically, in time, frequencies, and applications. One sample demonstrated configuration is illustrated in Fig. 12.

Each operator had an access to a repository of about fifty policies. An operator could selectively configure one or more radios she was permitted to control by changing configuration modes and by adding or removing one or more of the available policies from the radio's local repository. The interaction results were then displayed on the Test Console screens of particular radios as well as in the status window of the administrator's Console.

The demonstration of the policy components successfully showed that the policy-based approach is applicable and viable to software-defined radios. The prototype illustrated qualitatively the capabilities offered to radio to enforce policies. In combination with detecting mechanisms employed by XG-enabled software-defined radios, the approach ensures that software-defined radios do not cause harmful interference. Additionally, the policy-based approach showed the capabilities to manage radios and secure access control to interfaces changing the radio's configuration, thus addressing the second type of concerns. Consequently, the demonstration has successfully shown that a policy-based approach is applicable for responding to device malfunctions as well as to malicious users attempting to abuse software-defined radios.

## IV. Conclusions and Future Work

We designed and implemented a policy-based framework as part of the DARPA NeXt Generation communications program to study and address the security concerns raised with software-defined radios. The main emphasis of the framework is to overcome two types of concerns – harmful interference caused by a malfunctioning device and harmful interference caused by a malicious user.

In order to avoid a harmful interference caused by a malfunctioning radio, in tandem with signal-detection-based interference avoidance algorithms employed by XG-enabled software-defined radios, we designed a set of policy-based components that are tightly integrated with the accredited kernel on the device. The policy conformance and enforcement components are responsible for ensuring that a device does not

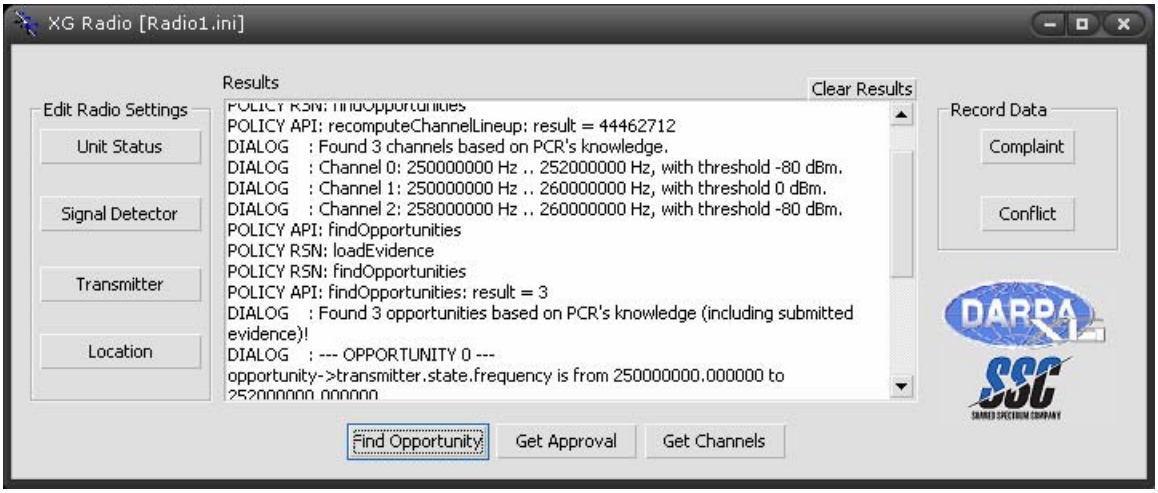


Figure 11. A sample screen snapshot of a Test Console for demonstrating the capability of policy-based control of software defined radios.

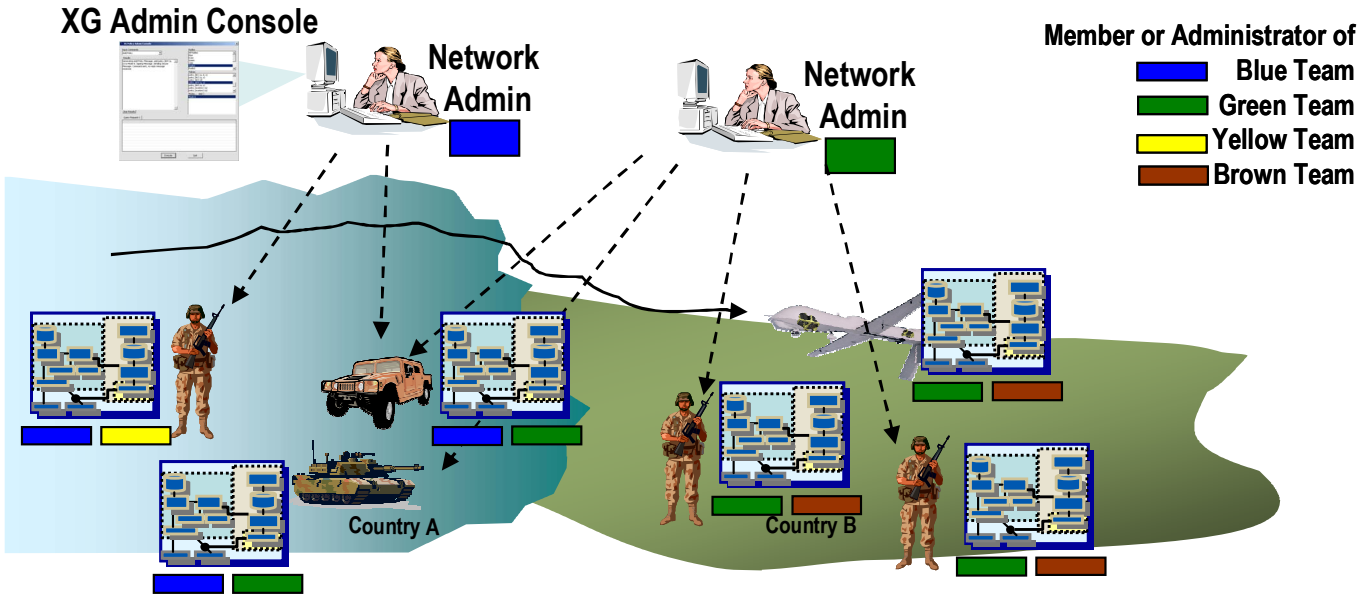


Figure 12. A sample scenario used to demonstrate the features and capability of a policy-based control of software-defined radios.

violate policies, which are encoded in a declarative language and which define regulatory and other stakeholders' requirements.

Since regulators and other stakeholders are not able to certify every state of a device, we've moved the certification and enforcement processes closer to the software-defined radio. The policy component represents an official regulatory proxy agent that enforces device's operations on behalf of the regulators based on their dynamic rules and requirements.

In order to avoid a harmful interference caused by a malicious user, we use secure policy management and distribution mechanisms in order to prevent malicious users from altering loaded policies as well as from inserting additional policies.

Suggested future includes: (1) Fielding this architecture in a deployed network to obtain additional requirements and detailed insights, and (2) Demonstrate that the architecture supports low cost, low processor power radios via a prototype hardware implementation.

## REFERENCES

[1] DARPA XG Program, http://www.darpa.mil/sto/smallunitops/xg.html, November 2006.

[2] F. Seelig and S. D. Jones, " A Description of the August 2006 DARPA XG Phase III Demonstrations at Ft. A.P. Hill", November 2006, *Submitted to conference.*

[3] Public-Key Infrastructure (X.509) (pkix) Charter, http://www.ietf.org/html.charters/pkix-charter.html.

[4] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference", W3C Recommendation, February 2004.

[5] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. "SWRL: A semantic web rule language combining OWL and RuleML", W3C Member Submission, 21 May 2004.

[6] Network Working Group, R. Enns, Ed., "NETCONF Configuration Protocol", Internet-Draft, draft-ietf-netconf-prot-12, February, 2006.

[7] T. Bray, J. Paoli, C. Sperberg-McQueen, and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C REC REC-xml-20001006, October 2000.

[8] J. Case, D. Harrington, B. Presuhn, and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2572, April 1999.

[9] Federal Communications Commission, "Rules to Permit Unlicensed National Information Infrastructure (U-NII) devices in the 5GHz band", FCC 03-287, November 18, 2003.

[10] OpenSSL: The Open Source toolkit for SSL/TLS, http://swi-prolog.org/, November 2006.

[11] J. Wielemaker, SWI-Prolog, http://openssl.org/, November 2006.

[12] SQLite 3, http://sqlite.org/, November 2006.