Extending Policy Languages with Utility and Prioritization Knowledge: The CAPRI Approach

Janne Riihijärvi*, Marina Petrova*, Vladimir Atanasovski† and Liljana Gavrilovska†
* Institute for Networked Systems, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
† Faculty of Electrical Engineering Ss. Cyril and Methodius, Skopje, Macedonia
email: jar@inets.rwth-aachen.de

***Abstract*—Cognitive Radio research has been focusing mostly on Dynamic Spectrum Access concepts, and one of the key architectural components there is the capability to express and regulate Policies. In this paper we will report on our work of extending the existing policy language and server concepts to a wider context. As cognitive radios will eventually become software defined radios, there is a number of non-spectrum related policies that require management. In fact, there is a rich interplay between spectrum and other policies, and they can influence each other in the final policy reasoning.**

We discuss the rationale on extending policy languages to include functional description of utilities that users and applications perceive. We add also various ways to express priorities and preferences between technologies and services. Further we report on the on-going work and results, which we have obtained on implementing such extensions as a part of CoRaL language that has been proposed earlier by SRI. We also outline policy implications of such extensions.

I. Introduction

Cognitive Radios (CRs) are capable of supporting Dynamic Spectrum Access. In wider perspective CRs are context sensitive smart radios that can adapt and optimize their performance. In both cases it is important that radios are policy aware, and support both permissive and restrictive policies. Spectrum Policy (SP) servers and languages have been studied extensively. Notable examples are the pioneering work done as a part of DARPA XG-project [1], and CoRaL language that has been developed and released by SRI [2].

The flexibility in the future radio systems is not limited only to the spectrum domain. Hence, there is a reason to consider if other policies should be also described and enforced through policy languages. Even more interestingly, the optimization and context based reconfiguration implies that cognitive radios must have goals and understanding of the cost functions that are related to these goals. Thus there is a need to express *utility functions* of different goals, so that the rational decisions can be made between different operating modes. Some earlier Quality of Service (QoS) description languages can be seen as partially relevant for this work, and utility based optimization of network is proposed earlier by [3].

In this paper we argue that keeping spectrum policy language separate from utility and priority (QoS) constraint descriptions may be artificial and inefficient. There is a rich interaction between these concepts, and at the reasoning level these different points of views must be taken into account anyway. We advocate on widening the perspective of the policy approach from narrow regulatory spectrum policy enforcement towards more general policy descriptions that include apart of regulatory policies also preferences and individual policies of technologies, users and other stakeholders. This sort of rich policy description language could provide new regulation possibilities, but opens also an avenue for new innovations in business and technology areas. Apart of describing our general approach to policies, we will ground some of our discussion by reporting the technical work we have done to implement prototype ontology and interfaces for such general approach. Moreover we will describe the architecture and protocol that has been developed to transmit and exchange policy descriptions between cognitive radios and policy servers. This work considers both earlier CoRaL language based spectrum policies and extension of it, as well as our Common Application Requirement Interface (CAPRI) [4].

The capability to describe utilities as a part of policy language can have also a strong impact on economics of dynamic spectrum assignment. It will also provide possibility to audit system goals.

This paper by necessity is positioned to be between policy and technology research. The rest of the paper is organized as follows. Section II describes our general approach and gives problem statements. In Section III, the system architecture and components are described. We also outline the extended policy language and report our early results on the wireless protocol to share policies. Section IV reports on related work, and in Section V we present our conclusions.

II. From Spectrum Policy to General Cognitive Radio Policy

A. Problem Statement

As discussed in the introduction, we consider cognitive radios (CRs) as devices capable of carrying out diverse *optimization decisions* in order to improve the Quality of Experience for their users. Performing dynamic spectrum access (DSA) decisions is one key enabler, but the overall scope of actions a CR might carry out is considerably wider, including, for example, selection of radio access technology, modulation order used, power allocation, and scheduling decisions related to different application flows. So far most of the policy work in CR space has focussed on the DSA case, namely expressing *constraints* on configurations CR is allowed to use on the physical layer. We argue that when treating CRs as devices carrying out optimization decisions this is not sufficient. First, even in the case of CR performing DSA only, it has to have *goals* it tries to achieve. These goals are clearly application dependent; for example a Voice-over-IP client has very different requirements compared to interactive web-browsing or watching a video from YouTube. Some of these goals can be expressed as constraints, fitting neatly into the usual framework of policy languages. However, in order to be able to reason about trade-offs between different settings when a diverse collection of applications is using the wireless link, more information is needed.

Second issue arises from the scope of the constraints. While constraints on spectrum access are clearly necessary, we believe they are not sufficient on the overall system level. For example, there are often regulatory constraints on the overall operation of the terminal relating to emergency calls, which yield overall constraints to the reconfiguration and optimization processes running on the terminal or the network. Also the users might want to impose constraints on the radio access technologies used for different applications, for example. Common choice amongst users is to transfer small payloads, such as email notifications, through cellular networks, but download large content only through free Wi-Fi hotspots. If the terminal or the network is to perform access selection for the user this should be formalized as a constraint on configurations allowable for the applications involved. These considerations yield the problem statement for the work presented in this paper: How should the current spectrum-related policy frameworks for cognitive radios be extended in order to yield both useful constraints and goals for reconfiguration and optimization?

B. What do we exactly do and propose?

We propose to build on the excellent work already carried out in the policy community in developing frameworks for expressing and reasoning about spectrum policies. We extend the scope of constraints from policy-related ones to more technical and economic ones. This requires extending the scope of the policy language used. Furthermore, we offer new interfaces for applications and the users to add new policies. In order to go beyond constraints, and also to include information needed to derive optimization goals, specific extensions are needed. The simplest way would be to take the approach originally adopted in cost functions for IP routing, namely attaching to each application a single *network attribute*, such as throughput or delay, that it would like the network to optimize the connection with respect to. However, this is rather limiting, and does not allow applications to have priorities or "weights" between each other. Instead, we propose to enable association of each application flow with a *utility function* which can be used directly as basis for diverse optimization decisions. Formally, utility function U is a function on some collection key network attributes $(a_1, \ldots, a_n)$, mapping each such collection to a real number characterising the "goodness" of the connection for the particular flow. Enabling specification of such functions requires a well-defined interface towards the applications, or towards a dedicated operating system component, and a data model and corresponding representation of the functions itself. These could be either integrated into an existing policy language framework that is sufficiently expressive, or treated separately. The former approach reduces the needed interfaces, whereas in latter case actual policy language parser and reasoner can be kept simpler, and support for utility functions made optional.

The utility descriptions also have applications beyond the node and network level scopes. One interesting avenue for further research is to use the utility descriptions to yield estimated *value* for spectrum, to be subsequently used by policy servers, spectrum auctioning processes, and so on. This would require propagation of the utility information through the network, and thus suitable protocol being developed, but these would be very straightforward extensions to the basic concepts introduced in this paper.

C. How do we approach the problem?

We have developed the above extensions using the well-known CoRaL policy language [2] as a foundation, and developed the required interfaces towards the applications for specifying additional policies and introducing utility functions for individual application components or data flows. Both of these functions are accessed through the CAPRI interface, but through separate function calls to maintain the modularity discussed above. Our extensions to CoRaL are performed using the inherent mechanisms present in the language, namely by introducing new *ontologies* for constraints relating to higher levels of the protocol stack and the reconfiguration and optimization processes themselves. No change to the actual language specification was necessary. For CAPRI we have specified a concrete language for describing utility functions [4], as well as defined a number of "helper functions" allowing developers to construct common examples of utility functions easily, without need to understand their detailed use in network optimization.

In addition to these extensions, we have developed a complete prototyping environment, including a distributed policy management and reasoning framework, and protocols for exchanging policy descriptions and results of reasoning processes on those policies. High-level components of the system are developed for standard Unix-like systems such as Linux, whereas standard software defined radio prototyping platforms (USRP2s and WARP-boards) are used as flexible radio access technologies in the testing and prototyping work. We believe experimental work is highly important, since that allows both concrete use cases to be studied, helping to identify the necessary functionalities, as well as enables evaluation of overhead and other key performance characteristics of the system.

In the following Section we shall give a detailed description of the implementation architecture of the developed components, and also describe some of the early evaluation results we have obtained.

III. Architecture and Implementation

In this section we discuss the architecture and implementation related issues for extending the scope of policies in cognitive wireless networks. We shall first discuss overall architectural requirements in some detail. This will be followed by a detailed discussion on integration of utilities and policies into the policy language and our CAPRI interface design as examples. Finally, we discuss in some detail our current prototyping activities towards enriched policy architectures for cognitive radios.

A. *Architectural Requirements*

In order to implement an approach integrating utility functions with a policy framework, there naturally needs to be a policy architecture in place. The policy architecture assumed in this paper is a very classical one, easily mapping to the numerous policy framework proposed in the literature. We assume the presence of two essential components every policy based cognitive radio system possesses, that is we assume there is a Policy Enforcing Point (PEP), which is used for network optimization through, for example, radio resource management, and a Policy Decision Point (PDP), for reasoning about policies, as described in [5], [6] (see Figure 1). In usual wireless network architectures the PEP would be a (cognitive) radio resource management entity, responsible for determining the state of the radio environment in which it operates and collecting information about spectrum usage, available channels and networks, location etc. Based on this information and the incorporated system strategies, the PEP creates policy requests (i.e. transmission and reconfiguration requests) which are sent to the PDP. Additionally, the PEP controls the transmissions and reconfiguration actions based on the received replies from the PDP.

The role of the PDP on the other hand is to perform the reasoning (i.e. decision) process. It is typically comprised of two entities, a policy database (DB) and a Policy Reasoner (PR). The policy database contains active policies in the system, while the policy reasoner performs reasoning for each PEP policy request individually [7], [8]. It checks the rules imposed by policies from the DB upon reception of a policy request and allows (if all policies allow it) or disallows (if at least one policy is violated) the policy requests. In case of a negative reply from the PR, the PEP would typically create a new policy (transmission) request based on previous experience and sends it again to the PR. The PEP must be present in every wireless node carrying out reconfiguration decisions, whereas the PDP is not mandatory for all nodes. However, the PEP could be implemented in a distributed fashion in centralized architectures such as in cellular networks. The core part of the PEP would in such systems reside in the radio network controller, which would then issue reconfiguration requests to

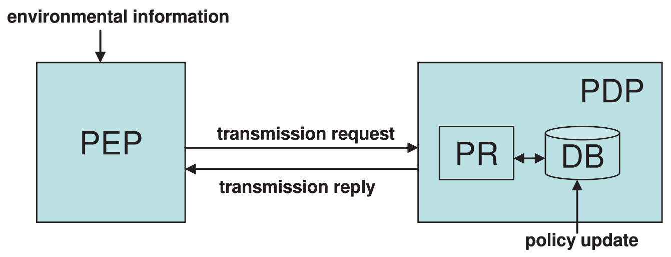

Figure 1. The general high-level policy architecture assumed in this paper.

base stations and terminals. The nodes without PR can use the reasoning capabilities of a remote neighboring node. The PDP is also referred to in the following text as a *Policy Engine* (PE) [9].

The policy system proposed in this paper is general and can support any reasonable policy language, e.g. spectrum, device, connectivity and QoS policies. These policies may originate from a regulator, an operator and/or a user. Spectrum policies are used to control the access to the spectrum by different users. They specify allowed frequency bands and upper limits for radiated power per device type and per frequency. A regulator usually supplies these constraints based on regional rules and international standards. Also, a network/service operator can supply policies in order to achieve traffic load balancing, to improve resource usage in the network or to enforce user prioritization. Device policies contain constraints related to device capabilities (e.g. maximum output power, processing time, supported application profiles etc.). They can be specified by the equipment vendor and the network (the traditional cellular network case) but also by the user, for instance to decrease the radiated power in order to save the battery life. Finally, connectivity policies contain rules based on user/operator preferences for establishing connections. These policies can specify e.g. the time intervals in which the services are allowed for usage by specific users. In this manner, connectivity policies provide a tool to control the network access. An association between the types of policies and their sources is given in [10].

In general policies can be static or dynamic. The policy is static when there is no policy change for a relatively long time period. If policy changes occur more frequently over time and locations the policy is sad to be dynamic. Regulator policies are usually more static than policies of operators and users. Operators and users policies are more dynamic since they can be changed with a faster rate.

In Figure 2 we present one possible logical decomposition of a policy based system from a global viewpoint. This framework comprises all policy relevant blocks and functionalities and introduces the need for a mechanism for policy distribution and loading policies into relevant databases. The policies coming from regulator and operator are stored into the corresponding central policy servers (Operators and Regulators policy servers) [11], [12], and are distributed into the databases of the local PEs.

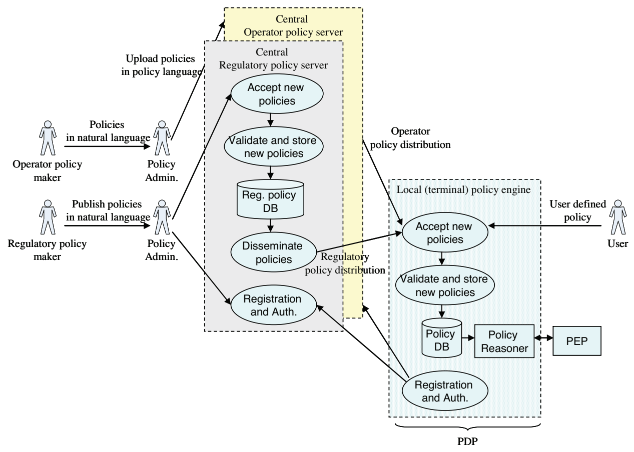

Figure 2. Logical view of a policy architecture.

B. Language

One of the hallmarks of the CoRaL language is that it has been carefully designed to be extendible, a feature we rely on heavily in our design. Extending the scope of the language is done by introducing new *ontologies*, defining new terms that can be used in policies and reasoned about. The default CoRaL ontologies cover spectrum-related policies very well, and have been sufficient for our work. For other policies defined by users and network operators we have defined ontologies for, e.g., link layer attributes and parameters and end-to-end connections. Following examples illustrate how these can be used in user-specified policies.

Our first example corresponds to the usage scenario involving costs of using different radio access technologies discussed above. For bandwidth-intensive but delay tolerant applications the user might want to impose a policy of using free connections only. This would be accomplished in our framework with the following policy specification passed as an argument to the CAPRI interface:

policy *free_connections_only* **is**
use *connection_attributes*;

disallow if
    connection.costPerUnit != 0;
end

The first line introduces the policy, and defines a name for it that can be used later for modifications or revocation. The second line introduces the new ontology for connections, which incorporates both attributes for radio access as well as end-to-end path. The actual condition of the policy is introduced in the **disallow if** statement, which disallows particular configuration if the monetary cost associated to it is nonzero.

In the second example we illustrate how constraints on performance can be expressed through the extended CoRaL

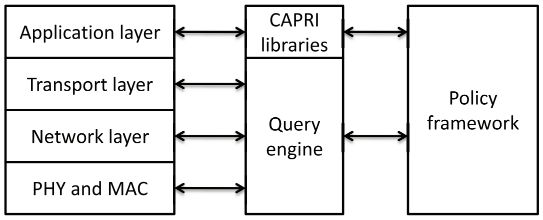

Figure 3. Relation between CAPRI and the policy framework.

policies. Recall that for most cases we expect applications to register utility functions that can be used in the network-wide optimization process. However, for some applications, such as VoIP clients with simple codec designs, a simple QoS-specification suffices. Following policy constrains used configurations to yield connections typically considered as yielding a good grade of service for VoIP:

policy *VoIP_good_grade* **is**
use *connection_attributes*;

disallow if
    connection.throughput $<$ 87.2 kbps **or**
    connection.delay $>$ 150 ms **or**
    connection.lossrate $>$ 0.03;
end

The structure of the policy is similar to the first example, only difference being the use of the throughput (in kbps), delay (in ms) and lossrate attributes being used in an inequality to specify disallowed configurations. Obviously, much more complex conditions could be specified as well. Essentially any common QoS specification can be expressed in the *connection_attributes* ontology, and using additional utility function specifications increases flexibility and richness of the available information significantly still.

While extending the language by introduction of new ontologies suffices for reasoning part, the overall software framework also requires small extensions. Key issue is interfacing towards the protocol stack to obtain current values for the attributes represented in the *connection_attributes* ontology. Our preferred solution is to isolate the logic required for this into a dedicated *query engine* which offers a simple database abstraction of the current attribute values towards the policy framework. Each ontology corresponds to a table in the abstract database, each attribute in that ontology to a column in the table, and, finally, each object such as connection to a row in a table. With such an implementation the policy framework and most of the query engine can be made both modular and highly portable, and platform and technology-specific components isolated as small extensions to the protocol stack. This overall implementation architecture is illustrated in Figure 3, showing the information flow and different interfaces in the developed software architecture.

For specifying utilities we use a simple dedicated language typical for computer algebra systems. The language used consists of integers and real numbers (in the commonly used dotted decimal or scientific notations), usual basic arithmetic operations (+ for addition, - for subtraction/negation, * for multiplication, / for division and for exponentiation) together with parentheses. Expressions $\log(a)$ and $\exp(a)$ are included, and evaluate to numerical approximations of the natural logarithm and exponential function with argument a. Additionally, the Iverson bracket notation is supported facilitating the expression of simple conditions, step functions etc. In this notation literal form $[\langle \textit{condition} \rangle]$ is used, evaluating to one if the $\langle \textit{condition} \rangle$ is satisfied, and to zero otherwise. For expressing the conditions the usual expressions $=$, $!=$, $<$, $<=$, $>$ and $>=$ are used. Finally, alphanumerical identifiers are used for the arguments of the utility function, corresponding to the various measurable attributes of the connection. At present these are "t" for throughput, "d" for delay, "plr" for packet loss rate, "ber" for bit error rate and "j" for jitter. In this language the utility function version of the good grade VoIP connection requirement expressed above in policy form would simply be

```
"[t > 87.2kbps]*[d < 150ms]*[plr < 0.03]".
```

However, to fully benefit from the power of the utility-based approach, it would be preferable to specify some objective measure of the VoIP call quality as the function of the connection attributes. Example of a metric yielding such a measure is the PQoS model introduced in [13], which can be defined in our framework as follows. First, we define a new attribute using one of CAPRI helper functions as

```
capri_define_attribute("R",
  "93.2 - (0.024 * d
    +0.11 * (d - 177.3) * [d > 177.3])
    -A*log(1+B*plr)-C"),
```

where `A`, `B` and `C` are codec specific constants to be set by the application. This defines the new attribute `R` expressed in terms of delay and packet loss rate which can now be used to define more complex utilities. The PQoS utility function (essentially in the 1–5 scale of the mean opinion score) is now given by

```
"[R <= 0] + [R > 0]*[R < 100]
  *(1 + 0.035*R+R*(R-60)*(100-R)*7e-6)
    + 4.5*[R >= 100]".
```

For further details, see [13]. Similar techniques can of course be applied to other applications as well, such as video streaming or interactive audio.

For applications of elastic type the utility functions usually have on the other hand logarithmic shapes with respect to throughput, given by the CAPRI specification `"A*log(t/B)"`. By changing the values of the constants `A` and `B` the user can prioritize flexibly between different applications, for example reducing capacity used by background tasks and increasing responsiveness of interactive applications running on the foreground.

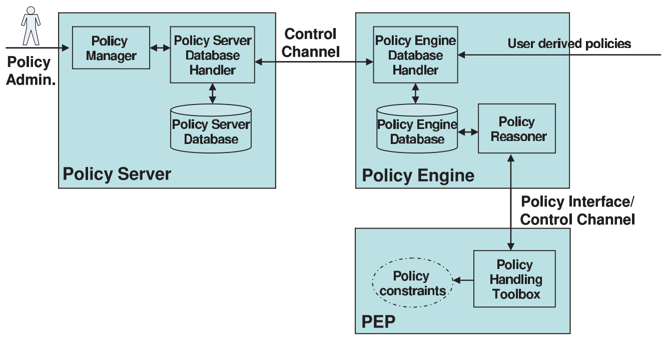

Figure 4. Policy system detailed view.

We are also currently working on operator and service provider specific ontologies further extending the scopes of the policies that can be expressed and used in the policy framework. These ontologies will consist of expressions of business relations, of payments either made or offered in spectrum auctions, of interference relations between systems and so on. The policy specification

policy *cheap_roaming_allowed* **is**
use *operator_contracts*;

allow if
(exists ?nw:network.list)
    homenetwork.audible(?nw) $= 0$;
(exists ?rc:roaming.contract)
    roaming.cost(?rc) < 0.15;
end

is a simple application example allowing cheap enough roaming provided that the home network of a terminal is not available and a contract exists between the two networks. Such extensions make it possible to define very rich interactions between the different stakeholders in a dynamic spectrum access scenario while using the same policy framework as a foundation. For example, one could expose utility definitions of users to the policy framework and extend the above example to also allow roaming should the expect utility while connecting to home network be too low. The implications of availability of such mechanisms are quite interesting. It would, for example, enable use of policies as business shaping tools, also potentially lowering the resistance of the incumbents in adopting more flexible spectrum access mechanisms since they would have a concrete platform where to express their concerns and requirements. This approach could also create new roles for regulators either as sources of commonly agreed upon policy definitions, or as auditors of policy frameworks to ensure correct operation and interoperability of solutions.

C. Prototype Architecture for Policy Engine

After discussing the essential building blocks and their interactions in a policy based system, we shall now discuss in more detail the specific implementation architecture of our policy framework prototype for cognitive wireless networks. Figure 4 depicts the basic elements and interfaces in the proposed policy system architecture. Our system has three main building blocks, namely Policy Server, Policy Engine (PE) and Policy Enforcing Point (PEP). The Policy Server is in charge of acceptance, updating, storing and disseminating the policies to the correspondent nodes. It consists of a Policy Manager (PM), a Policy Server Database Handler (PSDH) and a Policy Server Database (PSD). The PM receives the various policies generated by the policy administrators via a GUI and translates them into a policy language specific format. The PSDH is in charge of storing the policies into the servers' databases and disseminating them to the local policy engines. The PSD contains active policies and information about users in the network. It is organized in three parts, i.e. a user part containing information about active users, a policy part hosting policies and a part for associating the active users with corresponding policies. The user part includes the following fields:

1) User Name, containing the name of the user;
2) User ID, for active user identification;
3) User IP Address, containing the IP address of the user and used for sending policy messages to the corresponding users;
4) User Class, specifying the class of the user;
5) Device Type, specifying the class of the user's device.

The policy part includes the following fields:

1) Policy Name, specifying the name of the policy;
2) Policy ID, an ordinal number of the policy used for policy identification;
3) Policy Rules, containing the policy constraints expressed in a policy language syntax;
4) User Class, specifying the class of user for which the policy is dedicated. There can be different classes of users depending on their priorities. Some of the policies might be dedicated for all user classes.
5) Device Class, specifying the class of device for which the policy is dedicated. There can be different classes of devices depending on the device capabilities and radiated power.

The part for associating active policies with active users contains a User ID and a Policy ID field of the active users and policies, respectively.

The PE is a decision point in the policy architecture and is located in every PE equipped terminals. A PE is composed of a Policy Engine Database Handler (PEDH), a Policy Engine Database (PEDB) and a Policy Reasoner (PR). PEDH is used for accepting policies from Policy Servers and users and storing the policies into the PEDB. The PR performs the reasoning task deciding about each policy request from the PEP based on the active policies in the PEDB.

The Policy Handling Toolbox (PHT) is a part of the PEP which creates and sends policy requests to the PR. The PR checks all active policies from the PE DB and returns appropriate replies to the PHT. After receiving the approved transmission parameters, the PHT announces the policy constraints to the PEP and the PEP can control its transmission within the policy constraints.

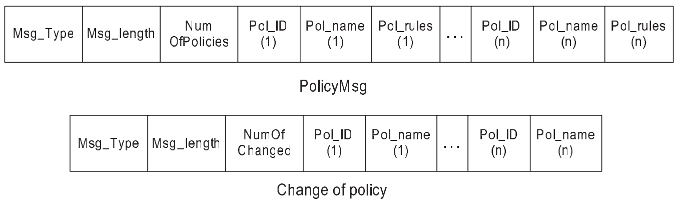

Figure 5. Policy messages.

We introduce two interfaces in our architecture, i.e. a policy interface and a control channel interface. The policy interface is used to handle policy request/replies between the PEP and the PR locally in every node with reasoning capabilities. The control channel interface is used to handle transmission among nodes for exchanging control information and for policy distribution from stakeholders (e.g. operator, regulator). It is important to mention that in case of remote reasoning, when the node without a PE uses the PE of a remote node, the standard policy request/reply messages are exchanged through the control channel.

D. Over the Air Policy Protocol

The policy framework architecture discussed above requires an over the air protocol for exchanging policy relevant information among architectural entities. It should be noted that the policy exchange protocol must provide way to both broadcast (or multicast) and unicast information to corresponding nodes. The policy protocol defines the communication between the PM and the PSDH, the PSDH and the PEDH and the PEDH and the PHT. This protocol is dedicated for management of policies, i.e. policy distribution, policy changes triggering, loading/updating of policies and policy reasoning. Messages defined with policy protocol are exchanged either through the control channel (in case of remote reasoning) or the policy interface (in case of local reasoning). All the message types of the protocol follow a common frame format that begins with *MsgType* field, used for distinguishing the messages, and *msg_length* field, indicating the length of the message in bytes. The subsequent bytes define content specific to the given message types.

The key message types are Policy Messages used to distribute policies to PE DBs, Policy Queries used for sending policy requests to the PR from the PHT, and Policy Replies returning the reply to the Policy Query from the PR. Additional message types are defined for signalling changes in active policies, for triggering policies related to emergency situations, for registering users and devices in the network to the policy framework, and for registering new types of policies into the system. Examples of the structure of these messages are shown in Figure 5. All of the protocol messages are encapsulated in TCP/IP packets and are sent through a TCP/IP socket connection between the communication points, unless only local (node-internal) communication is required. In the figure below we give couple of message examples.

One of the most active groups that investigate the field of policy based management and dynamic spectrum access is P1900.4 working group (active under IEEE SCC 41). The approach proposed in this paper shares some common goals with P1900.4, since both architectures aim to optimize radio network resource usage. However, there is a notable difference between them, because P1900.4 is network oriented whereas the one in this paper is terminal oriented [14].

E. Use Cases

Figures 6, 7 and 8 present several Message Sequence Charts (MSCs) illustrating the functionalities and the implementation aspects of the proposed policy based system through use cases. When a user attaches to a network, it announces its presence by sending registration message to the Policy Server. Afterwards, the Policy Server, based on the user's class and device type, selects the appropriate policies and sends them to the user. The user stores the received policies into its local database and senses the spectrum in order to find spectrum opportunities. After a connection request from the application is received, the PEP asks the PE for permission of transmission in a specific channel. If requested parameters for the transmission are in compliance with the policies in the local database, the reasoner returns affirmative reply, otherwise it denies the requested transmission and the PEP should find a new appropriate transmission parameters and request again. If some relevant policy is changed during the transmission, the policy server informs the nodes to stop transmitting and find a new channel. One other scenario is when radio environmental conditions are changed due to transmissions of other users. Then the user should change its transmission parameters, but first it must check the policy conformance. In such case, if policy rules are still satisfied it continues transmitting with the new requested parameters, else it stops transmitting and searches for a new channel opportunity. Figure 6 presents connection establishment between two users. After registration, the users get their dedicated policies from the Policy Server. After connection request is received, if the policies allow the requested transmission on both user sides, the communication can be successfully started.

Figure 7 depicts discontinuation of an established connection due to environmental changes. The PEP detects the changed environmental condition and calculates new transmission parameters, however the policy rules are not satisfied and the communication is therefore stopped. The communication is renewed after detecting and approving a new channel opportunity, thus demonstrating adaptive behavior of the users to the environmental changes.

Figure 8 illustrates connection interrupting due to changed policy on the policy server. In this case, the PR sends triggered negative policy reply, so the ongoing transmission is stopped. The communication between the two end-nodes continues when a new parameters configuration is successfully approved and established.

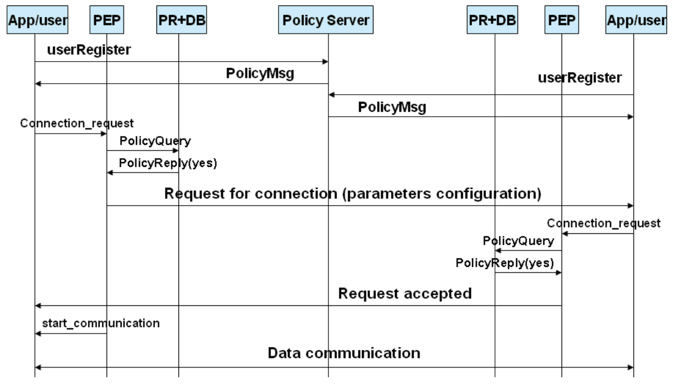

Figure 6. Establishing connection.

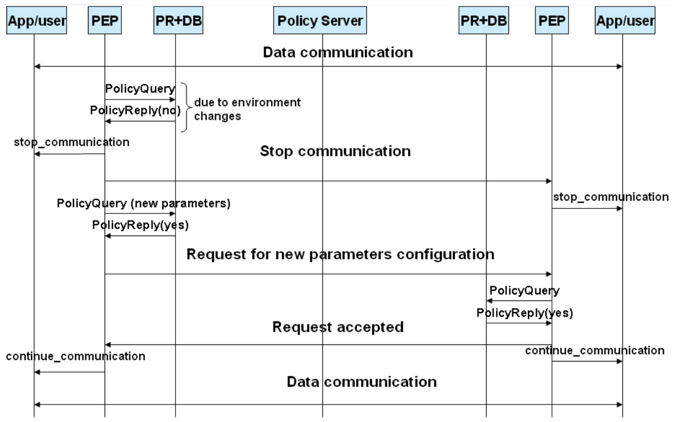

Figure 7. Connection interruption due to environmental changes.

The elaborated scenarios in this subsection can find practical implementation examples in cognitive environments. For instance, when the terminal moves to other geographical location it downloads the policies specific and valid for that location. Thus, in different geographical zone, different policies will define the behavior of the terminals. The second use case is typical example that when channel conditions are degraded because of primary user appearance, the cognitive radio device should change the channel. The last use case may have wider meaning. The policies can be changed either because of the balancing the traffic load from an operator or for providing QoS to a specific class of users or because the policy validity expires after certain time interval etc.

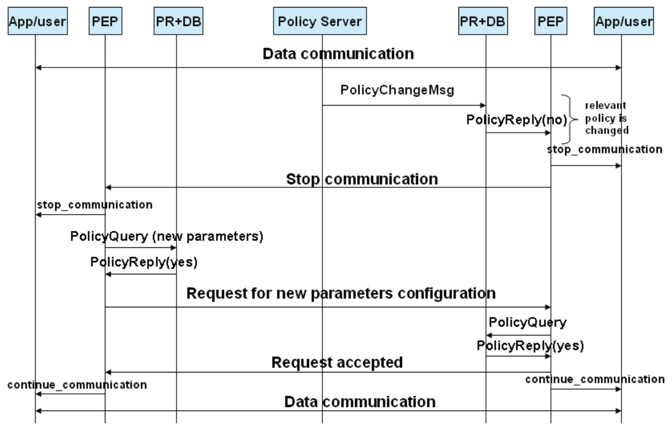

Figure 8. Connection interruption due to policy changes.

IV. Related Work

Policy languages and frameworks have been extensively studied especially in the dynamic spectrum access domain. We have already mentioned the work at SRI towards the CoRaL language [2] as well as the work in the DARPA XG working group as important building blocks [1]. Earlier influencal policy language designs in related domains have been [15] and [16], whereas Mitola's Radio XML [17] and its predecessor the Radio Knowledge Representation Language (RKRL) [18] embed policy descriptions into a more general knowledge representation framework. See also [19] for a related discussion.

Policy languages can also assist in providing the necessary crisp and precisely defined ontologies for discussing and reasoning about regulatory problems and interactions between different stakeholders in DSA scenarios. Lack of such a precise terminology, and potential pitfalls of reasoning with loose terminology, has been pointed out by, for example, de Vries in [20]. Catering for these needs is another important reason for exploring the space of cognitive radio related ontologies more carefully.

Regarding more general policies discussed here and on mechnisms for specifying optimization goals for the network, key examples of earlier work have been developed in the context of the wired Internet. Using policies to impose constraints on routing decisions is the most common example, and at heart of the Border Gateway Protocol (BGP). See [21] for a broad overview. Similarly early work on QoS-related objectives and policies has been in the routing domain [22]–[24]. Using utility functions as a general framework for understanding and reasoning about network design and operation has been a more recent development, some of the key references being [3], [25]–[27].

Finally, the developments reported on here are tightly coupled to more general architectural considerations related to cognitive radios and dynamic spectrum access. It is precisely in richer spectrum sharing architectures such as the DIMSUM-Net design of Buddhikot et al. [28] that extensions of policy languages as described here can be utilized in full. Indeed, we expect that such extensions might in fact be necessary for the successful adoption of such a cognitive wireless network architecture.

V. Conclusions and Future Work

The experimentation we have done with the developed architecture and protocol show that it is possible to exchange even relatively detailed policy descriptions over the present day wireless broadband networks. Thus the overhead of going beyond the simple spectrum access policies does not seem to be an issue. Extending the policy language has potentially profound implications for users, service providers and regulators

First, our proposed approach with policy languages allow end-user generated (limited) policies. This provides users, and notably cognitive radios themselves, a capability to express their wishes, constraints and perceived utilities. One should note that although this enables better user-centric approach, it is also beneficial for cognitive network operators, since this sort of information allows better optimizations towards user preferences. The use of policy languages and our CAPRI interface provides more flexible and natural way to integrate this information to DSA and CR environments than using separate QoS mechanisms.

As discussed a sufficiently rich policy language would be beneficial for operators. They could control better the use of their networks. Since they could within limits set up flexibly policy constraints for cooperation and opportunistic use, it would lower the barrier to accept changes in the market place. Again the constraint statements and utility function descriptions fit also perfectly to different economic incentives, including using them as a part of possible dynamic spectrum auctioning mechanisms. Our developed policy exchange protocol supports also this sort of use well, since we can choose to whom we would be passing policy information.

Our specific and extended policy language might have an intriguing implications also in regulatory domain. As the policy languages are by necessity very precise, machine readable and based on crisp ontologies they might be highly interesting for the regulators. The current paradigm has been to see policy languages mostly as the tool to express regulators rules. Our extension is bringing also other players of the value chain into the play. Thus one could speculate that regulators would have interest to obtain also full policy definitions for their scrutiny. This would provide a transparent way to audit that rules are followed, and to understand different interplays between partners and competitors in the market. This does not necessarily mean loss to operators, but one can see policy descriptions as fully auditable way to prove, e.g. existence of competition and non-existence of cartels. This is not as speculative as it might sound, one good example is the recent push by European Union competition and anti-trust regulators on understanding and limit roaming prices in the cellular networks. The regulatory domain is even more complex in the case of the future DSA and CR based networks. This example also underlines the fact that regulations are not limited to spectrum regulations, nor they are limited to single countries or single regulatory agency. If DSA becomes popular and mainstream technology, especially through different spectrum leasing and sharing methods, the competition agencies and regulators will become legitimately interested in. We argue that it might be worthwhile already to provide mechanisms that enable competition and also remove doubts from those agencies.

Another lesson from our current work has been that although a lot of work can be done in the theoretical domain, one needs to build also experimental systems to understand better the implementation limitations of different approaches. Although laboratory implementations cannot test the rich dynamics of real environment and markets, it allows at least limited testing and finding out which implementation methods support best a chosen regulatory enforcement methods.

One of the main messages of this paper is to advocate both policy and technology communities to consider policies and policy languages as general tools. The community can enable a lot of new innovation, if we dare to look beyond narrow scope of spectrum policy languages. Our preliminary work in this domain shows that there are a number of interesting extensions that can be done. We will not necessarily want to push any of our own ontologies and provided extensions, but urgently advice to debate on the opportunities of extended policy languages.

Acknowledgment

The authors would like to thank RWTH Aachen University and the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) for providing financial support through the UMIC research centre. We would also like to thank the European Union for providing partial funding of this work through the ARAGORN and FARAMIR projects.

References

[1] DARPA XG Working Group, “The XG Vision. Request for Comments, BBN Technologies, Cambridge, MA,” July 2003.

[2] D. Elenius, G. Denker, M. Stehr, R. Senanayake, C. Talcott, and D. Wilkins, “CoRaL–Policy Language and Reasoning Techniques for Spectrum Policies,” in *Proc. of the 8th IEEE International Workshop on Policies for Distributed Systems and Networks*. IEEE Computer Society Washington, DC, USA, 2007, pp. 261–265.

[3] S. Shenker, “Fundamental design issues for the future Internet,” *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[4] J. Riihijärvi, M. Petrova, and P. Mähönen, “A common application requirement interface for cognitive wireless networks,” in *Proc. of the 4th Annual IEEE Workshop on Network Technologies for Software Defined Radio (SDR) Networks, in conjunction with SECON 2009*, Rome, Italy, June 2009.

[5] S. Boros, “Policy-based network management with SNMP,” *CTIT Technical Report Series*, no. 00-16, October 2000.

[6] D. Wilkins, G. Denker, M.-O. Stehr, D. Elenius, R. Senanayake, and C. Talcott, “Policy-based cognitive radios,” in *IEEE Wireless Communications Special Issue on Cognitive Wireless Networks*, 2007.

[7] K. Amiri, S. Calo, and D. Verma, “Policy based management of content distribution networks,” in *IEEE Network Magazine*, March 2002.

[8] M. Ayari, F. Kamoun, and D. Males, “Towards a policy-based management for ad hoc networks,” in *IFIP Med-Hoc-Net’04*, June 2004.

[9] T. C. Clancy, Z. Ji, B. Wang, and K. J. R. Liu, “AI planning approach to dynamic spectrum access in cognitive radio networks,” in *Proc. IEEE Global Telecommunications Conference (Globecom) 2007*, November. 2007.

[10] L. Berlemann, S. Mangold, B. Walke, and R. ComNets, “Policy-based reasoning for spectrum sharing in radio networks,” in *Proc. of 1st IEEE DySPAN 2005*, 2005.

[11] R. Chadha, C. Chiang, M. Little, and S. Samtani, “Agent-based policy-enabled network management architecture for mobile ad hoc networks,” in *Proc. IEEE Military Communications Conference*, October 2003.

[12] M. Ahmed, V. Kolar, M. Petrova, P. Mahonen, and S. Hailes, “A component-based architecture for cognitive radio resource management,” in *4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2009)*, June 2009.

[13] H. Koumaras, F. Liberal, and L. Sun, “PQoS Assessment Methods for Multimedia Services,” *Handbook of Research on Wireless Multimedia: Quality of Service and Solutions*, p. 316, 2008.

[14] S. Buljore et al., “IEEE P1900.4 System Overview on Architecture and Enablers for Optimised Radio and Spectrum resource usage,” in *Proc. IEEE Dynamic Spectrum Access Networks conference (DySPAN) 2008*, 14th 17th October, 2008.

[15] L. Kagal, T. Finin, and A. Joshi, “A policy language for a pervasive computing environment,” in *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, 2003.

[16] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, “KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement,” *Policy*, vol. 93, 2003.

[17] J. Mitola, *Cognitive radio architecture: the engineering foundations of radio XML*. John Wiley and Sons, 2006.

[18] J. Mitola and G. Maguire, “Cognitive radio: making software radios more personal,” *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.

[19] D. Lewis, K. Feeney, K. Foley, L. Doyle, T. Forde, P. Argyroudis, J. Keeney, and D. O Sullivan, “Managing policies for dynamic spectrum access,” *Lecture Notes in Computer Science*, vol. 4195, p. 285, 2006.

[20] J. de Vries, “Imagining radio: Mental models of wireless communication,” in *Proc. of 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, DySPAN 2007*, April 2007, pp. 372–380.

[21] O. Younis and S. Fahmy, “Constraint-based routing in the internet: Basic principles and recent research,” *IEEE Communications Surveys and Tutorials*, vol. 5, no. 1, pp. 2–13, 2003.

[22] Z. Wang and J. Crowcroft, “Quality-of-service routing for supporting multimedia applications,” *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[23] S. Shenker, C. Partridge, and R. Guerin, “Specification of guaranteed quality of service,” *RFC 2212*, September 1997.

[24] S. Chen and K. Nahrstedt, “An overview of quality of service routing for next-generation high-speed networks: problems and solutions,” *IEEE Network*, vol. 12, no. 6, pp. 64–79, 1998.

[25] M. Chiang, S. Low, A. Calderbank, and J. Doyle, “Layering as optimization decomposition: A mathematical theory of network architectures,” *Proceedings of the IEEE*, vol. 95, no. 1, p. 255, 2007.

[26] D. Palomar and M. Chiang, “A tutorial on decomposition methods for network utility maximization,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1439–1451, 2006.

[27] J. Lee, M. Chiang, and A. Calderbank, “Utility-optimal random-access control,” *IEEE Transactions on Wireless Communications*, vol. 6, no. 7, p. 2741, 2007.

[28] M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans, “DIMSUM-Net: New directions in wireless networking using coordinated dynamic spectrum access,” in *Proc. of IEEE WoWMoM05*, 2005, pp. 78–85.