An implementation of collaborative adaptation of cognitive radio parameters using an ontology and policy based approach

Shujun Li, Jakub Moskal, Mieczyslaw M. Kokar & David Brady

Analog Integrated Circuits and Signal Processing
An International Journal

ISSN 0925-1030

Analog Integr Circ Sig Process
DOI 10.1007/s10470-011-9681-y

Springer

Analog Integr Circ Sig Process
DOI 10.1007/s10470-011-9681-y

An implementation of collaborative adaptation of cognitive radio parameters using an ontology and policy based approach

Shujun Li · Jakub Moskal · Mieczyslaw M. Kokar · David Brady

Received: 22 February 2011 / Revised: 2 June 2011 / Accepted: 25 June 2011

Abstract The main objective of this paper is to demonstrate the feasibility of the ontology and policy based approach to realize collaborative, automatic adaptation of cognitive radio parameters in the transmitter and receiver. The ontology and policy based approach features the following three aspects. First, all the internal and external information of the radio is represented in the Web Ontology Language (OWL). The transmitter and receiver share the same core ontology and thus can communicate with each other using this common language. Second, the policy rules are written in a declarative form and interpreted by a reasoner. Third, in order to exchange the OWL represented information between the radios, we adopt a more flexible signaling plan, which is different than the conventional protocol-predefined signaling plan, i.e. the control messages are added to an extendable payload, rather than embedded in the predefined protocol-dependent header or trailer. The paper discusses the implementation and shows some examples of radio behaviors resulting from the execution of the policies.

Keywords Software defined radio · Cognitive radio · Ontology · Semantic web · Policy · Adaptation · GNU radio

S. Li (✉) · J. Moskal · M. M. Kokar · D. Brady
Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA
e-mail: li.shuj@neu.edu

J. Moskal
e-mail: jmoskal@ece.neu.edu

M. M. Kokar
e-mail: kokar@coe.neu.edu

D. Brady
e-mail: d.brady@neu.edu

1 Introduction

Cognitive radio [13, 14, 19] is expected to have the capabilities to (1) sense the environment and collect information of the environment; (2) be aware of the external situation, the internal state and its own capabilities; (3) automatically adapt its objectives; (4) reason about communications situations, objectives and radio configurations. Some of these capabilities, such as spectrum sensing and opportunistic utilization, are currently actively pursued by various wireless research projects. However, the capabilities of awareness and reasoning, especially combined with adaptation, have not been previously reported in the literature.

Awareness, including situation awareness and self-awareness, is the ability to interpret, or "understand", the input information [9, 19]. More specifically, it refers to the perception of the elements in the surrounding environment, the comprehension of their meaning and the projection of their status in the near future [4]. Real awareness can be achieved only if the agent can reason about the facts it gets from the environment or from other agents. Reasoning refers to the ability to infer implicit knowledge from the explicitly represented knowledge. Reasoning requires (1) a proper language to represent the knowledge and policies, and (2) a reasoning engine that can process the knowledge and policies.

Awareness and adaptation are the most critical requirements incognitive radio. Due to the growing complexity and heterogeneity of the communication networks, cognitive radio needs to be aware of the changing context and adjust the services and resources accordingly. The changing context includes the spectrum environment, the user and demands, the network topology, etc.

However, there are a few limits in the current radios that hinder them to fulfill awareness and adaptation: (1) Local

Published online: 20 July 2011

information is stored in a data model that does not have high expressivity and machine interpretable semantics. For example, in SNMP (Simple Network Management Protocol) or CMIP (Common Management Information Protocol), the information that can be stored and retrieved is limited to scalar variables. (2) In the protocol-based communication, signaling messages are limited by the frame structure defined by the protocol, and therefore hinder the ability to express more complicated messages. For instance, it is not possible to exchange complicated information such as the structure of a component. (3) XML technology can be integrated with the existing protocols to provide a means to express more complicated information [1, 16]. However, XML-represented information cannot be processed by an inference engine since XML lacks formal semantics. Therefore, cognitive radios cannot reason about an XML representation unless explicit information interpretation procedures are written beforehand.

A new approach is required to replace the current static design solution with a more flexible and adaptable solution, e.g., a policy based solution. Also, the radios need to use a language that all of them "understand", so that they are capable to exchange and process information about their situations, goals, components and capabilities [8].

Various academic communities and wireless research projects are actively working towards such a goal. In [18], the authors proposed the concept of Ontology-Based Radio (OBR). In the OBR approach, all the internal/external information and the signaling messages are represented in the Web Ontology Language (OWL). OWL is a formal language with high expressivity and computer interpretable semantics and therefore is capable of expressing complicated information and can be processed by an inference engine. Bearing the same concept, the Modeling Language for Mobility (MLM) working group in the Wireless Innovation Forum is leading an effort to develop a formal language, with computer interpretable semantics, that can be used to describe all aspects of network operations and management [8, 20, 2, 21]. Papers [5] and [10] discuss the language issues that arose in the process of developing the ontology and policies for cognitive radio. In [11], a public safety use case was used to demonstrate how to combine ontology, policy and an inference engine to control the radio behavior. In addition, the IEEE P1900.5 working group is working on a specification for a policy language for expressing and reasoning about spectrum access policies.

Aligned with the above efforts, this paper aims to show the feasibility of the ontology and policy based approach to realize automatic adaptation of cognitive radio parameters. We test it on a link adaptation use case. The contributions reported in this paper include the following: (1) we developed a Cognitive Radio Ontology in OWL to represent the basic terms of wireless communications; (2) based on this ontology, we developed a set of policies and rules to optimize the link performance with a goal of maximizing the power efficiency; (3) we implemented the ontology and policy based approach on a GNU/USRP radio platform.

In the rest of this paper, Sect. 2 describes a refinement of the OBR architecture. Section 3 introduces the concepts of inference engine, policy and ontology. Section 4 gives a brief description of the link adaptation problem along with the policies we developed for this use case. Section 5 shows the validity of the link adaptation policies through MATLAB simulations. Section 6 gives some details of the implementation of the ontology and policy based approach on the GNU/USRP radio platform and demonstrates some adaptation results. Section 7 concludes the paper and outlines future work.

2 Conceptual architecture

There are two types of parameters in a cognitive radio: *knobs* and *meters*. Knobs refer to the adjustable parameters that control the radio's operation and thereby affect the radio performance. Meters refer to the utility or cost functions that are intended to be maximized or minimized in order to achieve optimum radio operation.

In order to enable two radios to negotiate about their knobs and meters, we propose a refined architecture of the OBR as shown in Fig. 1.

Radio Platform provides the digital signal processing and software control, as well as the interfaces to communicate with the RF, sensors, information source/sink and the policy reasoner.

System Strategy Reasoner (SSR) is an internal component of the cognitive radio. It forms strategies to control the operation of the radio. The strategies reflect the opportunities, capabilities of the radio and waveform, and the needs of the network and users [3].

All the incoming messages from the RF are first processed by the Radio Platform. Data messages are passed to the radio application (we call it Data Sink), whereas the control messages end up in the SSR. Similarly, all the outgoing control messages are generated by the SSR and then passed to a buffer. The data messages and control messages are merged in the buffer and then passed to the Radio Platform. After being processed in the Radio Platform, the messages are sent out through the RF channel.

There are different models to describe the control mechanism of the OBR. Figure 2 shows an example of the closed loop feedback control model [6]. SSR can be viewed as a controller. The controller calculates the knobs as a function of the goal and the observed meters. Then, the plant (Radio

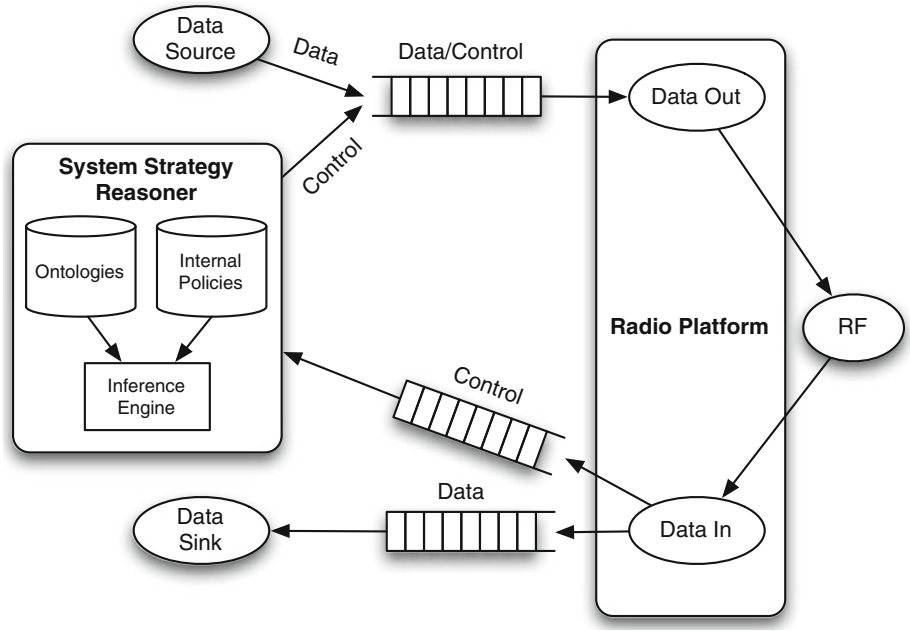

Fig. 1 A refined architecture of ontology–based radio

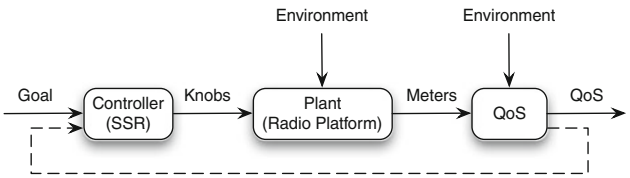

Fig. 2 Radio control model

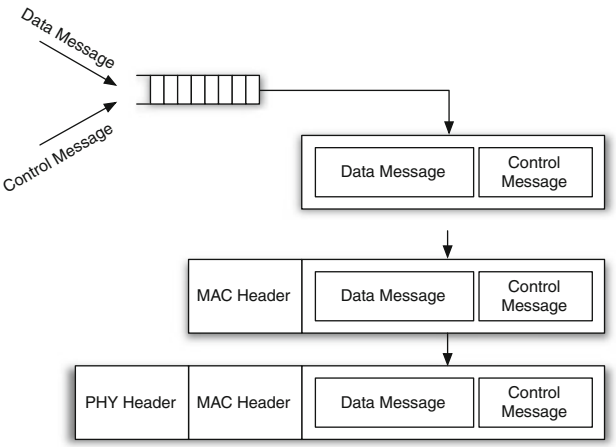

Fig. 3 Extensible payload field

Platform), being the actual operational part of the radio, collects the knobs, the meters and other sensed information from the environment and computes the Quality of Service metric (QoS). The QoS reflects the overall performance of the system. The QoS is sent to the controller (SSR) as a feedback. The controller then evaluates whether the goal is achieved. If not, then the controller changes its input to the plant (knobs) as to achieve the control goal [7].

As shown in Fig. 3, control messages are added to an extensible payload field as needed, rather than embedded in the predefined protocol-dependent header or trailer. This mechanism not only adds signaling flexibility to the existing protocols, but also makes it possible to add the inference capability to a radio without much change of the lower layer architecture.

3 Ontology and policy

SSR provides an inference capability to OBR. As shown in Fig. 1, SSR consists of an inference engine, an ontology and a set of policies. When the radio starts functioning, the ontology, the policies and the dynamic facts generated by the radio are loaded into the inference engine. Then the inference engine infers new facts and generates control messages such as a request to change parameter values.

3.1 Ontology

An ontology is a formal, explicit specification of a set of concepts in a specific domain and the relationships between these concepts. The term "formal" means that the ontology is machine interpretable for the purpose of knowledge reuse and sharing [17]. We developed a Cognitive Radio Ontology to cover the basic terms of wireless communications from the PHY and MAC layers. The ontology is written in OWL and includes the following top-level classes: (1) Object (2) Process, (3) Quantity, (4) Value, and (5) Unit of Measure. The relationships among the top-level classes are shown in Fig. 4.

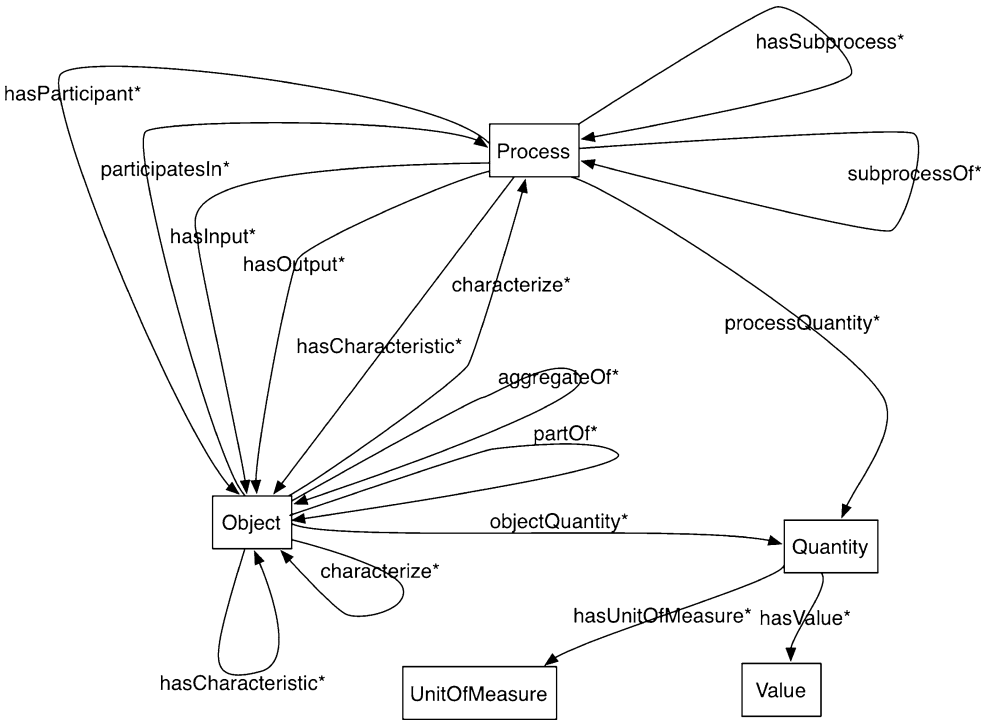

Fig. 4 Top-level classes of cognitive radio ontology

Object refers to the entity that is wholly presented at any given snapshot of time. Examples include material objects such as a piece of paper or an apple, and abstract objects such as an organization or a law [12].

Conversely, *Process* is the entity that can be presented only partly at any snapshot of time. A process can have temporal parts and spatial parts. For example, the first movement of a symphony is a temporal part of a symphony, whereas the symphony performed by the left side of the orchestra is a spatial part of a symphony. In both cases, a part of a process is also a process itself [12].

Qualities are the basic attributes or properties that can be perceived or measured. Qualities cannot exist on their own; they must be associated with either an object or a process. All the qualities have values and some qualities have unit. The qualities without units are represented as data-type properties. The qualities with units are associated with a type of quantity [12].

Quantity is a representation of a property of an object. In other words, quantity is a representation of quality. For instance, a physical quantity represents a property of a physical object. Quantity carries three types of information: the type of the quantity (e.g., mass, length), the magnitude of the property (typically a real or integer number) and the unit of measurement associated with the given magnitude (e.g., [kg], [m]). In this ontology, quantity is a top-level class; it is further divided (sub-classified) into different types, such as length, frequency, time, etc. Each quantity is associated with a unit and a value.

Note that there is no explicit Quality class in our ontology. Instead, we use *object Quantity* and *process Quantity* to represent the quality of an object or a process. There are two perspectives to represent the quality of an object (or a process) depending on whether the quality has a unit or not. If the quality has a unit then the quality is represented as a sub-property of object-type property *object Quantity*. If it does not have a unit then it is represented as a data-type property.

The most basic relation that links objects and processes is *participate In*. An object cannot be a part of a process, but rather participates in a process. For example, a person is not a part of running, but rather participates in running.

Another basic relationship is aggregation. An object can be *partOf* (or *aggregateOf*) another object; a process can be *subProcessOf* (or *hasSubprocess*) of another process.

Besides the above two relations, we have also identified other basic relations among the top-level classes. For instance, the input and output of a process are objects. The capabilities of an object are a collection of processes. The characteristics of an object or a process can be represented as objects.

In summary, the Cognitive Radio Ontology has 230 classes and 188 properties. The documentation of this ontology is published in [21] and [10]. Using the classes

and properties defined in this ontology, we can develop policies to control the behavior of the radio.

3.2 Policy

A policy is a set of rules written in a policy language. In our implementation, we use BaseVISor as the inference engine. The policy rules are expressed in the BaseVISor syntax (BVR). The basic structure of BVR is RDF triple. An RDF triple consists of a subject, a predicate and an object. The subject and the object denote resources (things in the domain of discourse), and the predicate denotes a relationship between the subject and the object. BaseVISor is a forward-chaining rule engine optimized for handling facts in the form of RDF triples. The engine also supports XML Schema Data Types.

The following is an example of a rule in the BVR form:

```
<rule name = "checkPerformance">
  <body>
    <triple>
     <subject variable = "X"/>
     <predicate resource = "rdf:type"/>
     <object resource = "rad:SignalDetector"/>
    </triple>
    <triple>
     <subject variable = "X"/>
     <predicate resource = "rad:signalToNoiseRatio"/>
     <object variable = "SNR"/>
    </triple>
    <lessThan>
     <param variable = "SNR"/>
     <param rdf:datatype = "xsd:float">15</param>
    </lessThan>
    <greaterThan>
     <param variable = "SNR"/>
     <param rdf:datatype = "xsd:float">10</param>
    </greaterThan>
  </body>
<head>
  <assert>
    <triple>
     <subject variable= "X"/>
     <predicate resource = "rad:performance"/>
     <object
rdf:datatype = "xsd:string">acceptable</object>
    </triple>
  </assert>
</head>
</rule>
```

This rule states that if the SNR is smaller than 15 and larger than 10, then the performance is acceptable. In the BaseVISor syntax, the subject, predicate and object element can be a resource, a XML data type or a variable. If an element is a resource, e.g. SignalDetector, then this element is defined in the ontology.

As is the case with some inference engines, it is possible to extend the BaseVISor functionality by adding new functions. These are called *procedural attachments* or *functions*.

The following is an example of computing an objective function.

```
<bind>
  <param variable = "objFunc_PowdB"/>
  <computeObjFunc>
     <param variable = "PowdB_new"/>
     <param variable = "trainPeriod"/>
     <param variable = "m"/>
     <param variable = "v"/>
  </computeObjFunc>
</bind>
```

In this example, the objective function *<computeObjFunc>* is a user-defined procedural attachment. It has four arguments. It returns the value of the objective function and binds the value to variable *objFunc_PowdB*.

4 An example: link adaptation

In this paper, we use a link adaptation use case to show how OBR can be used to achieve automatic adaptation in cognitive radio. In this use case, there is a transmitter–receiver pair. The general goal of link adaptation is to maximize the information bit rate per transmitted watt of power, subject to a set of constraints. This is attained by fine-tuning the parameters in the transmitter and the receiver. Here, the OBR provides a means to exchange the control messages between the transmitter and the receiver.

The tunable parameters (knobs) and the observable parameters (meters) of the transmitter and receiver are shown in Table 1.

4.1 Objective function

The goal is to maximize the information bit rate per transmitted watt of power as shown below. In the following, all the parameter names are terms from the ontology.

Table 1 Parameters of transmitter and receiver

Tx		
PowdB	Transmission Power (Unit:dBm)	Knob
trainPeriod	Length of training sequence (Unit: channel symbol)	Knob
m	Index of (2^m-1, 2^m-1-m) Hamming Code	Knob
v	Integer of QAM modulation, 4^v is the size of QAM constellation	Knob
Payload	Size of payload field. Set payload = 128 bytes	Fixed
FracSpacing	Number of samples per symbol. Set fracSpacing = 2	Fixed
SampleRate	Number of samples per second. Set sampleRate = 1000	Fixed
Rx		
M	Number of feedback taps	Knob
N1	Number of precursor feedforward taps	Knob
N2	Number of postcursor feedforward taps	Knob
mSNR	Mean Signal-to-Noise Ratio	Meter

$$\text{InformationBitRatePerTxPower} = \frac{\text{BitsOfPayload/Time}}{\text{TxEnergy/Time}} = \frac{\text{BitsOfPayload}}{\text{TxEnergy}} = \frac{\text{payload} \cdot 8}{\text{TxEnergy}}$$

The transmission energy (Unit: Joules) equals to:

$$\text{TxEnergy} = \text{TxPower} \times \text{Time} = \frac{10^{\frac{PowdB}{10}}}{1000} \cdot \frac{\text{TotalNumberOfSymbol} \times \text{fracSpacing}}{\text{Sample Rate}}$$

The total number of symbols in the packet is:

$$\text{Total Number of Symbol} = [(32 + 8 \cdot \text{payload}) \cdot (1 + \frac{m}{(2^m - m - 1)}) \cdot \frac{1}{2v} + \text{trainPeriod}]$$

Assume that $payloadsize = 128$, $SampleRate = 1000$, and $fracSpacing = 2$ are fixed. Then the goal is to minimize the following objective function:

$$\text{objFunc} = 10^{\frac{PowdB}{10}}[\frac{528 \cdot (1 + \frac{m}{2^m - m - 1})}{v} + \text{trainPeriod}]$$

4.2 Constraints

The following are the constraints in this adaptation problem. These constraints must be maintained by the policies.

(1) The equalizer *mSNR* must be between 10 and 15 *dB*. Intuitively, a value greater than 10 *dB* yields good detection performance, but a value greater than 15 *dB* indicates that the data rate could be increased, or the transmit power should be decreased. Hence, the constraint for *mSNR* is:$10\ dB \leq mSNR \leq 15\ dB$.

(2) *PowdB* is the transmission power in *dBm*. Here, we set the upper bound of *PowdB* as: $PowdB \leq 0\ dB$. Since both *PowdB* and *mSNR* are in *dB*, a drop of *PowdB* results in an equal drop in *mSNR*. Thus, $\Delta PowdB = \Delta mSNR$.

(3) m controls the coding overhead of $(2^m-1, 2^m-1-m)$ Hamming code. m does not affect the *mSNR*. m has the natural lower bound of 3 and has no natural upper bound. Here, we set the constraint of m as: $3 \leq m \leq 7$.

(4) v controls the size of the QAM modulation. The natural lower bound of v is: $v \geq 1$. Increasing v by one unit drops the *mSNR* by approximately 6 *dB*. Thus, $\Delta v = \Delta mSNR/6$.

(5) *trainPeriod* affects the *mSNR* in a less clear way. If *trainPeriod*is less than $5 \cdot (M + N1 + N2)$, then the equalizer does not fully converge, thus *mSNR* will be decreased. If *trainPeriod* is greater than $5 \cdot (M + N1 + N2)$, then it will have little effect on *mSNR*, but will work against the minimization of the metric. Hence, the constraint of*trainPeriod*is: $5 \cdot (M + N1 + N2) \leq trainPeriod \leq 10 \cdot (M + N1 + N2)$.

(6) M, $N1$, $N2$ have a threshold influence on *mSNR*: the *mSNR* will increase with M, $N1$, $N2$, until a sufficiently large equalizer for the multipath is achieved. After that point, increasing the equalizer dimension will have no effect, except for increasing the shortest possible training sequence.

4.3 Policies

4.3.1 Policies for establishing the communications link

The link adaptation is accomplished by executing the policy rules for which the pre-conditions are satisfied. Figure 5 shows an example of such executions.

Suppose we initialize radio A as the transmitter and radio B as the receiver. After radio B receives a data message from radio A, radio B invokes its reasoner. Then a rule is fired to check the performance by measuring the *mSNR*. Radio B then first sends a "query" message to radio A, asking for the current values of its parameters. When radio A receives this query, it invokes its reasoner, which decides whether it can accept this query. If yes, then radio A sends an "agree" message to radio B, followed by an "inform-ref" message that contains the answer to the query. After radio B receives the answer, its policy rule is triggered to compute the new values of its local parameters and the parameters of radio A. Then radio B generates a request to radio A. The "request" message contains the

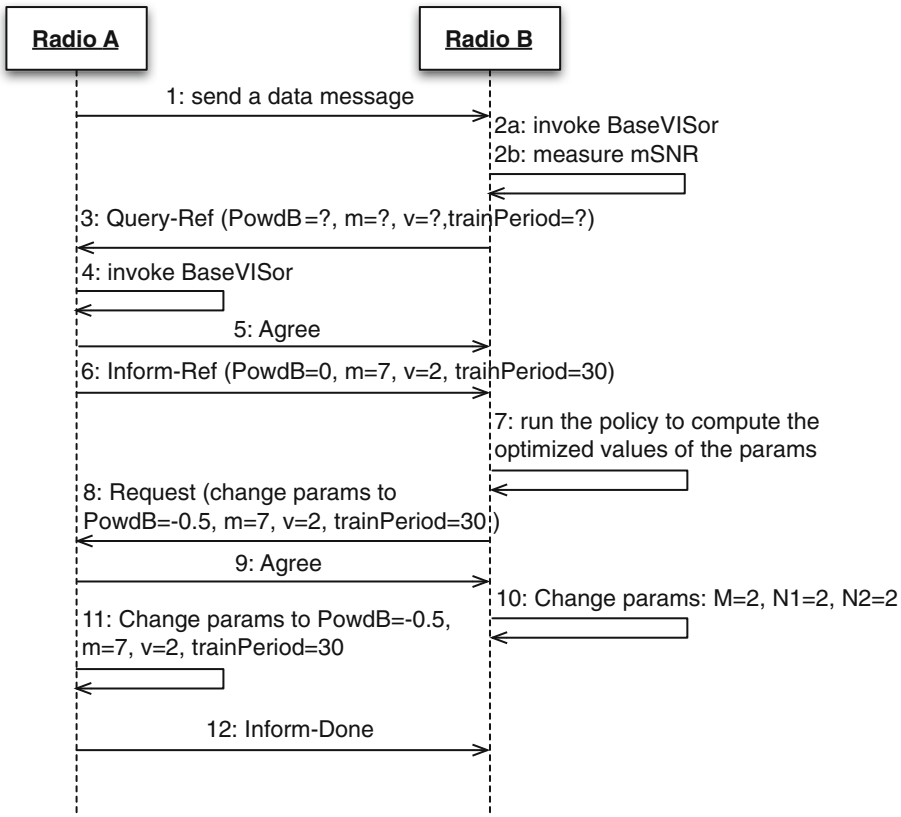

Fig. 5 Sequence diagram of link optimization

new values of radio A's parameters. After radio A receives the request, it runs its reasoner to decide whether it can accept this request. If yes, then radio A sends an "agree" message to radio B and then sets its parameters accordingly. After radio A finishes setting its parameters, it sends an "inform-done" message to radio B.

The following is an example a rule (in pure text) for reacting to the received message:

Rule "checkPerformance":

If the radio receives a data message, then (1) check mSNR; (2) generate a query message; (3) send the query to the originating radio.

4.3.2 *Policies for link adaptation*

The goal of link adaptation is to minimize *objFunc*. However, the decrease of *objFunc* will worsen the performance and decrease the *mSNR*. In other word, there is a tradeoff between the decrease of *objFunc* and the improvement of *mSNR*.

We implemented three sets of policies with different preferences. Policy1 decreases the *objFunc* as much as possible while not guaranteeing *mSNR* within the acceptable range. Policy2 decreases *objFunc* to an intermediate level while maintain *mSNR* in the acceptable range, if possible. Policy3 decreases *objFunc* while guaranteeing the *mSNR* within the acceptable range.

The following shows the description of Policy3; it contains four rules:

Rule 1:

If mSNR $>$ 15, then tune M, N1, N2 as follows:

$\Delta M = -2$, $\Delta N1 = -2$, $\Delta N2 = -2$.

Rule 2:

If mSNR $>$ 12.5, then tune one of these parameters: PowdB, trainPeriod, m or v as follows:

(1) Compute the following:

PowdB_new = min((12.5 − mSNR + PowdB), 0)

trainPeriod_new = min(7.5 *(M+N1+N2), trainPeriod)

m_new = 7

v_new = max(v, floor((mSNR − 12.5)/6)+v)

(2) Compute the following objective function values:

objFunc(PowdB_new, trainPeriod, m, v)

objFunc(PowdB, trainPeriod_new, m, v)

objFunc(PowdB, trainPeriod, m_new, v)

objFunc(PowdB, trainPeriod, m, v_new)

(3) Choose the smallest objective function value from (2) and tune the corresponding parameter to the new value.

continued

Rule 3:

If mSNR <= 12.5, then tune one of these parameters: PowdB, trainPeriod, m or v.

(1) Compute the following:

PowdB_new = min((15 - mSNR+PowdB), 0)

trainPeriod_new = min(10 *(M+N1+N2), trainPeriod)

m_new = 0

v_new = max(v, floor((mSNR-15)/6)+v)

(2) Compute the following objective function values:

objFunc(PowdB_new, trainPeriod, m, v)

objFunc(PowdB, trainPeriod_new, m, v)

objFunc(PowdB, trainPeriod, m_new, v)

objFunc(PowdB, trainPeriod, m, v_new)

(3) Choose the smallest objective function value from (2) and tune the corresponding parameter to the new value.

Rule 4:

If mSNR < 10, then tune M, N1, N2 as follows:

$\Delta M = +2$, $\Delta N1 = +2$, $\Delta N2 = +2$.

All the above policies are represented in the BVR syntax.

5 Simulation in MATLAB

In order to evaluate the validity of the policies, we simulate the link adaptation in MATLAB. We use MATLAB to emulate a Rayleigh multipath channel between two radios. Each radio is connected to BaseVISor. When a new message comes in, BaseVISor is invoked. The outputs of BaseVISor include new values of the parameters for the next transmission.

In order to evaluate whether the policy is able to adapt to the change of the channel environment, we linearly increase the number of multipath from 2 to 16. Assume the radios are operating in half-duplex mode. The default parameter values are: $PowdB = 0$, $m = 3$, $v = 1$, $trainPeriod = 100$, $M = 2$, $N1 = 2$, $N2 = 2$. First, we set the number of multipath to 2. Then radio A sends the first data message to radio B. When radio B receives the first data message, it measures *mSNR* and *objFunc*. Based on the current values of *mSNR* and *objFunc*, the two nodes follow the steps described in Fig. 5 to compute the parameters for the second data message and then set their parameters to the new values. Then we change the channel environment by setting the number of multipath to 4. After that, radio A sends the second data message to radio B and repeats the above steps. In total, radio A sends eight data messages to radio B. The simulation results and the comparison of these three policies are shown in Fig. 6. It can be seen that without link adaptation, *objFunc* remains at the same value and *mSNR* fluctuates as the number of multipath changes. With link adaptation, *objFunc* is significantly decreased. Policy 1 decreases *objFunc* by 66% at the price of decreasing *mSNR* by 1.83 dB. Policy 2 decreases *objFunc* by 55% at the price of decreasing *mSNR* by 0.83 dB. Policy 3 decreases *objFunc* by 36% while increasing *mSNR* by 0.09 dB.

6 Implementation on GNU/USRP

To further assess our ontology and policy approach, we implemented the link adaptation on GNU/USRP radios. GNU Radio is a free software development toolkit that provides the signal processing blocks to implement software radios using external RF hardware and processors. The Universal Software Radio Peripheral (USRP) is a high-speed USB-based board that enables general-purpose computers to function as software radios. In order to transmit and receive RF signal, the USRP motherboard is connected to daughter boards. The daughter boards are used to hold the RF receiver and transmitter. In our implementation, we used USRP1 as the motherboard and RXF2400 as the daughterboard. RXF2400 daughterboard is operated in the RF range from 2.3 to 2.9 GHz.

6.1 Implementation architecture

The implementation architecture is shown in Fig. 7; it is an extension of the conceptual OBR architecture shown in Fig. 1.

The *CORBA ORB and Service* is the middleware that enables the GNU Radio to act as a CORBA server and provide clients (upper layer applications) with means to transmit and receive data using the callback mechanism.

The *LiveKB* component provides a generic GET/SET API, which allows the reasoner to access and adjust radio's parameters. The details of LiveKB are discussed in [15].

Monitor Service (MS) is responsible for passing control messages between SSR and Data In/Out (DI/DO). When a control message comes in, DI will pass it to MS. MS unwraps the outer part of the control message and passes the content to SSR. The content is written in OWL/RDF and thus can be processed by the inference engine. The outer part of the control message specifies the type of the control message and is defined using the FIPA ACL message structure. The Foundation for Intelligent Physical Agents (FIPA) is a non-profit organization that develops specifications supporting interoperability among agents and agent-based applications. The FIPA Agent Communication

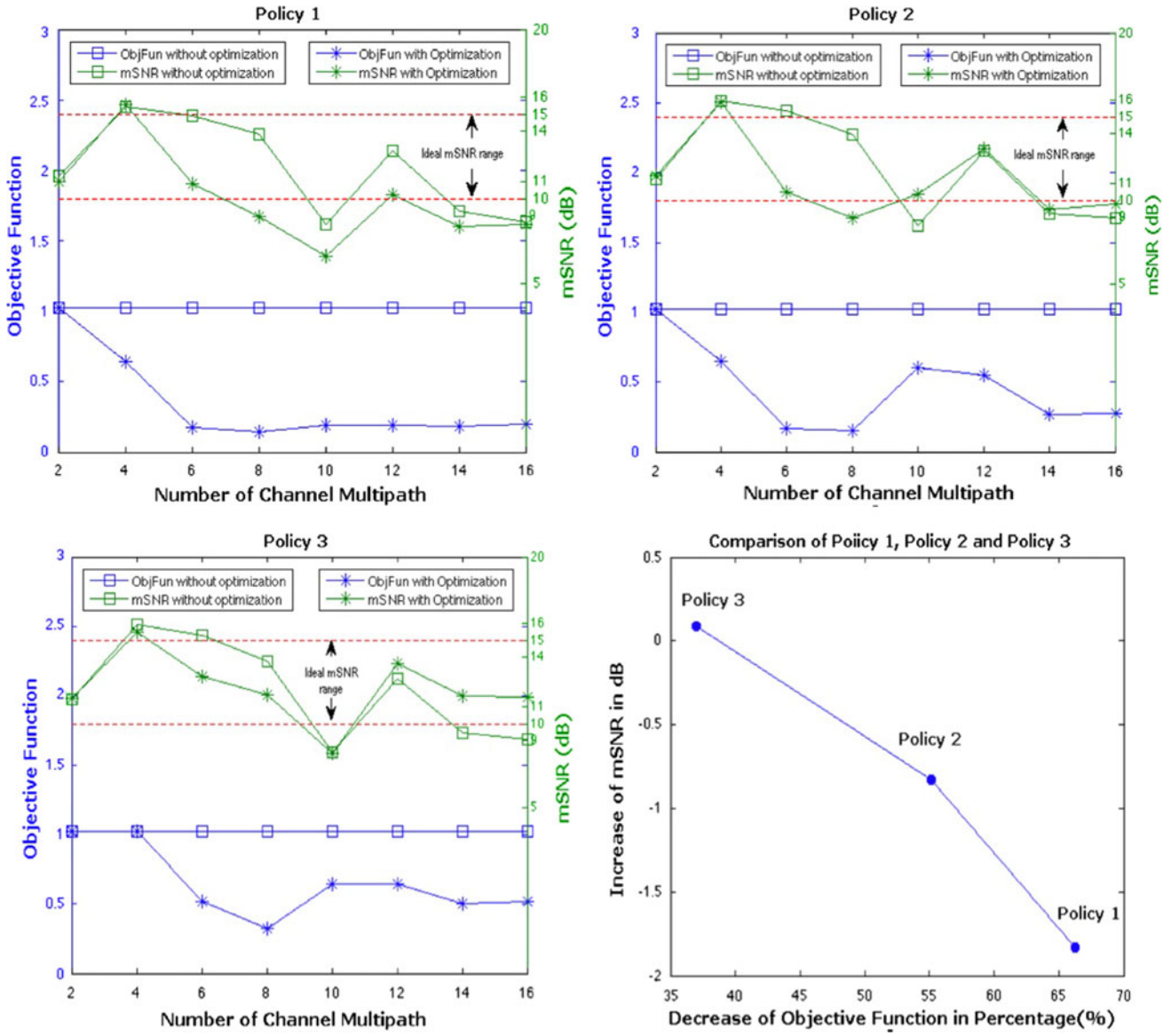

Fig. 6 Comparisons of policy 1, policy 2 and policy 3

Language (ACL) provides a standard set of message structures and message exchange protocols. The details of the FIPA ACL message structure will be discussed in Sect. 6.2.

After SSR receives the content from MS, it will start the reasoning. Figure 8 shows an illustration of how the two radios share the same knowledge base. T-Box contains the basic terms of the domain and includes the definitions of classes and properties as defined in the Cognitive Radio Ontology. R-Box contains the policies/rules specified in a declarative form, describing how to react to different situations. A-Box contains the facts that are only available when the radio is operating; they are the instances of the classes in the T-Box and are generated by the system in run-time.

6.2 Message structure and state machine

In our implementation, we adopt the FIPA ACL specifications to construct the control messages and design the finite-state-machine of the MS component.

6.2.1 Message structure

A control message contains two parts: (1) a set of message parameters, and (2) the content. The message parameters

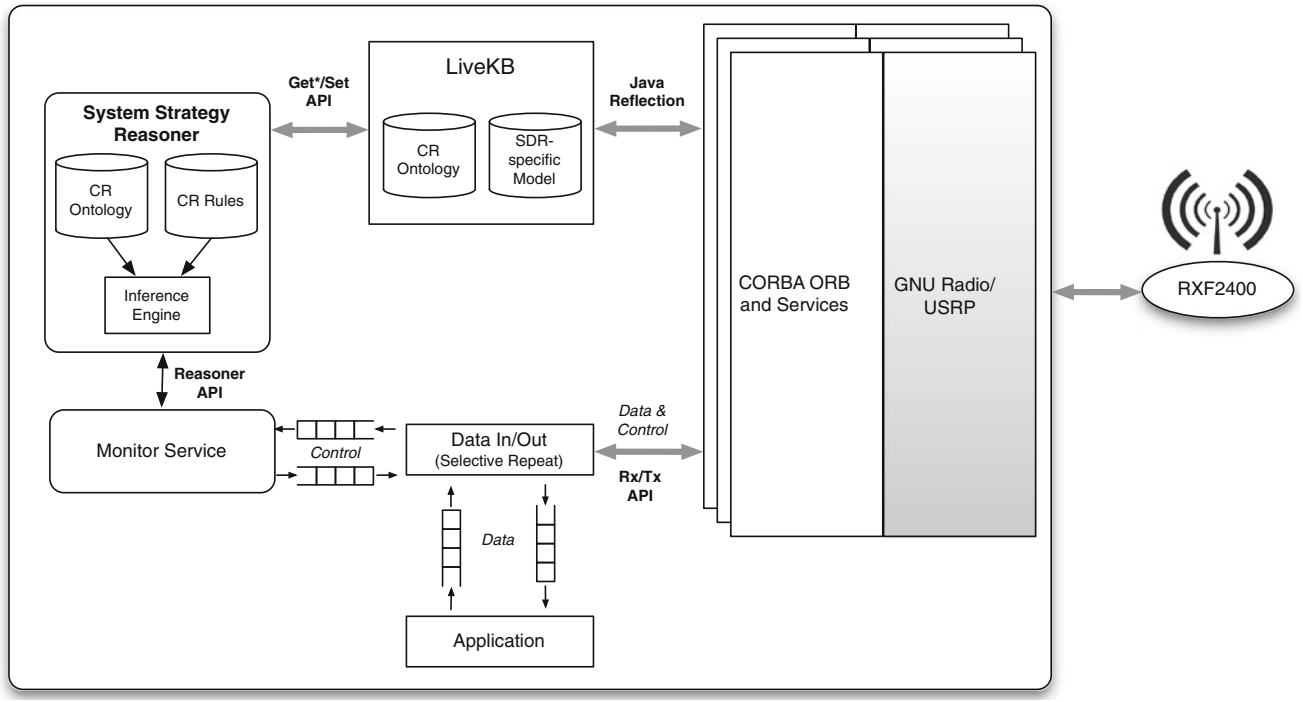

Fig. 7 Implementation architecture

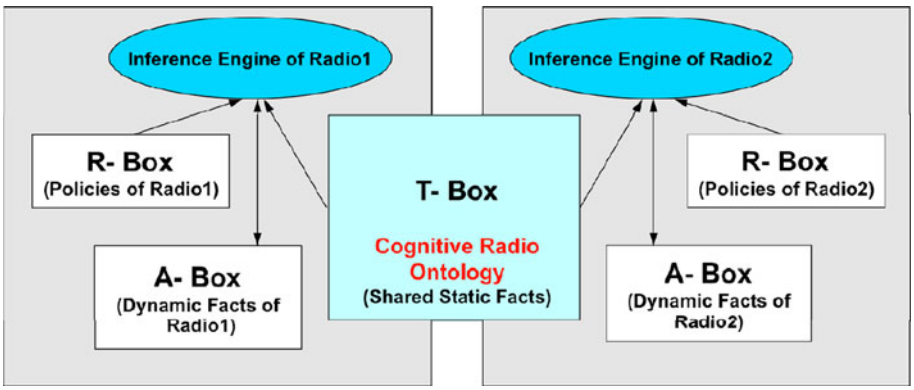

Fig. 8 Illustration of shared knowledge

provide information such as the type of message, the sender and receiver, the conversation id, etc. The content is described using a FIPA ACL content language. The choosing of a content language depends on the user's need. In our implementation, we chose OWL/RDF as the content language because it has the machine interpretable syntax and can be directly processed by the inference engine. The FIPA ACL specification defines 22 types of control messages, shown in Table 2. The sequence diagram shown in Table 2 is an example of two radios interacting with each other using some of these control messages.

The following is an example of a "request" message saying that radio B requests radio A to change its transmitter amplitude to 0.31.

Table 2 Types of control message

1	Accept proposal	12	Propagate
2	Agree	13	Propose
3	Cancel	14	Proxy
4	Call for proposal	15	Query If
5	Confirm	16	Query Ref
6	Disconfirm	17	Refuse
7	Failure	18	Reject proposal
8	Inform	19	Request
9	Inform If	20	Request when
10	Inform Ref	21	Request whenever
11	Not understood	22	Subscribe

```
(REQUEST
    :sender(agent-identifier:name radioB)
    :receiver(set(agent-identifier:name radioA))
    :content
     "<?xml version=\"1.0\"encoding=\ "utf-8\"?>
    <root>
      <triple>
        <subject resource=\"Run\"/>
        <predicate resource=\"FIPA\"/>
        <object resource=\"Request\"/>
      </triple>
      <rule name=\"request-from-radioA\">
       <body>
        <triple>
         <subject resource=\"Run\"/>
         <predicate resource=\"FIPA\"/>
         <object resource=\"Request\"/>
        </triple>
       </body>
       <head>
         <println>Changing tx_ampl to
0.30912052540030427 </println>
         <set>
         <param>/sdro:Radio/sdro:participate
sIn/sdro:hasParticipant/sdro:txAmplitude
         </param>
         <param datatype=\"xsd:float\">
         0.30912052540030427</param>
         </set>
       </head>
      </rule>
    </root>"
)
```

If radio A gets the above request message and decides to accept this request, it sends an "agree" message back to radio B as shown below:

```
(AGREE
      :sender(agent-identifier:name radioA)
      :receiver(set(agent-identifier:name radioB))
      :reply-with radioB1295829968769
)
```

6.2.2 *Finite state machine of monitor service*

As it was mentioned in Sect. 6.1, Monitor Service is responsible for processing the outer part of the control message and then passing the content to the SSR. Since FIPA ACL specification already provides the protocols to support the message interactions between radios, we can design the finite state machine of the Monitor Service based on the provided protocols.

Recall the scenario shown in Fig. 5. At step 8, radio A receives a request from radio B and then runs its inference engine to decide whether or not to accept the request. If the request conflicts with the local regulations (e.g. the transmitter power is out of the permitted range), then radio A will send a "refuse" to radio B. However, the two radios still have the chance to negotiate with each other until an agreement is met. According to the protocol provided by FIPA ACL, after radio B receives the "refuse" message, it can send a "call for proposal" to radio A. Then radio A can make a proposal to radio B. If the proposal is accepted, then radio A can change it parameters to the proposed values. Figure 9 shows the sequence diagram of the scenario.

The finite-state-machine implemented in Monitor Service for the Call-For-Proposal (CFP) scenario is shown in Fig. 10. It corresponds to steps 10 through 15 in Fig. 9. Note that the "initiator" in Fig. 10 refers to radio B and the "participant" refers to radio A.

6.3 Policy execution results

The results of policy execution are shown in Fig. 11.

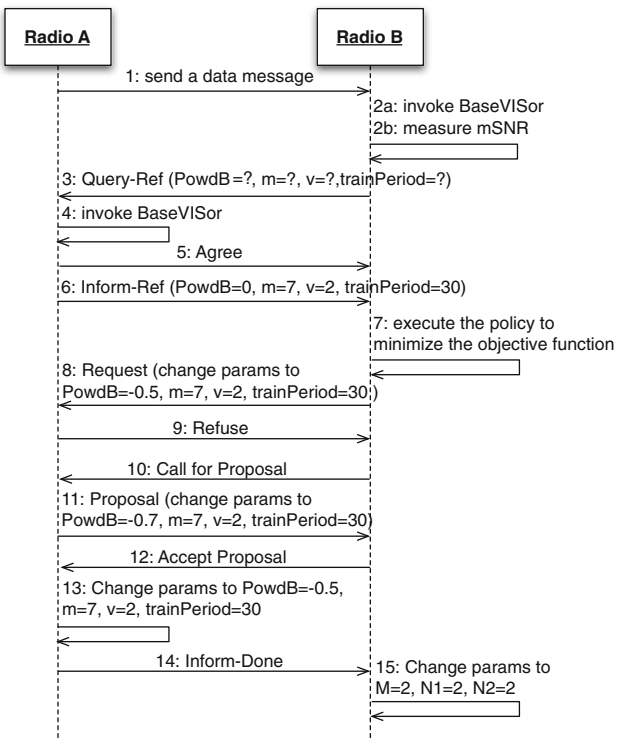

Fig. 9 Sequence diagram of call-for-proposal

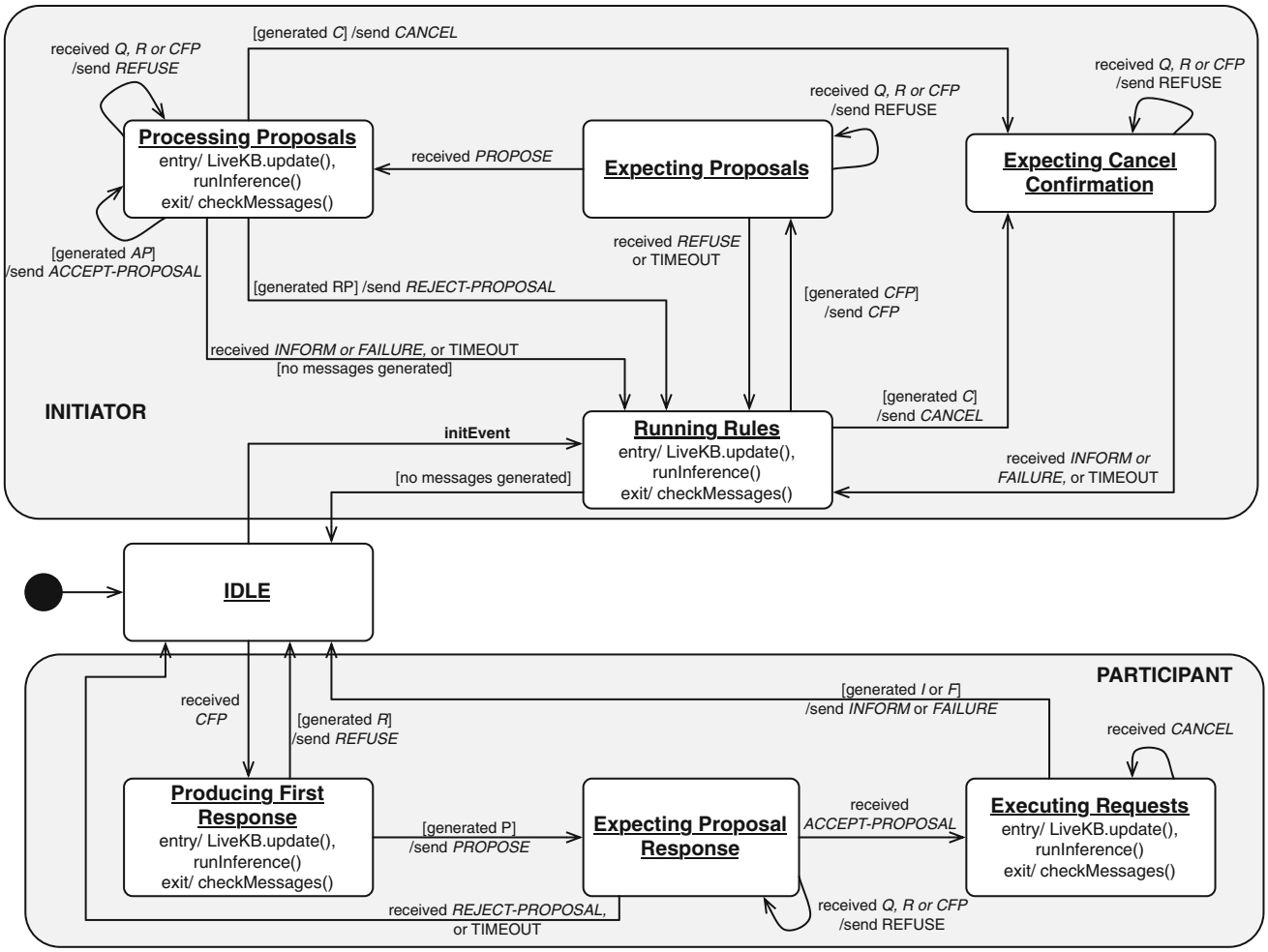

Fig. 10 Finite-state-machine of call-for-proposal

In the experiment, radio A keeps sending an image to radio B. Radio B measures the *mSNR* and then initiates the link adaptation process. The horizontal axis at the bottom shows the number of packets received. The vertical axis on the left (red-line) shows the value of the objective function. The vertical axis on the right (green line) shows the *mSNR* measured at radio B. Note that the objective function used in the implementation is the power efficiency, i.e., the information bit rate per transmitter watt of power (in$Gbit/watt \cdot sec$); it is the reciprocal of the objective function that is used in the MATLAB simulation. During the experiment, we moved the two radios around and deliberately changed the distance between them. It resulted in some changes of the channel environment and thus some fluctuation of the *mSNR*. It can be seen that when *mSNR* is too large, the radios adjust their parameters to lower the *mSNR* and thus increase the power efficiency. When *mSNR* is too small, the radios adjust their parameters to increase the *mSNR* for the price of lowering the power efficiency.

7 Conclusions and future work

In this work, we refined the OBR concept and showed the feasibility of implementing the ontology and policy based approach on the GNU/USRP radio platform. The goal is to realize automatic adaptation of cognitive radio parameters and thus improve the link performance, e.g., the power efficiency. Using our proposed architecture, the ontology and the rules we developed, the two radios are capable of negotiating their knobs and meters to achieve the goal In the future, we will further assess the costs and benefits of the ontology and policy based approach: how much benefit can we get from the use of

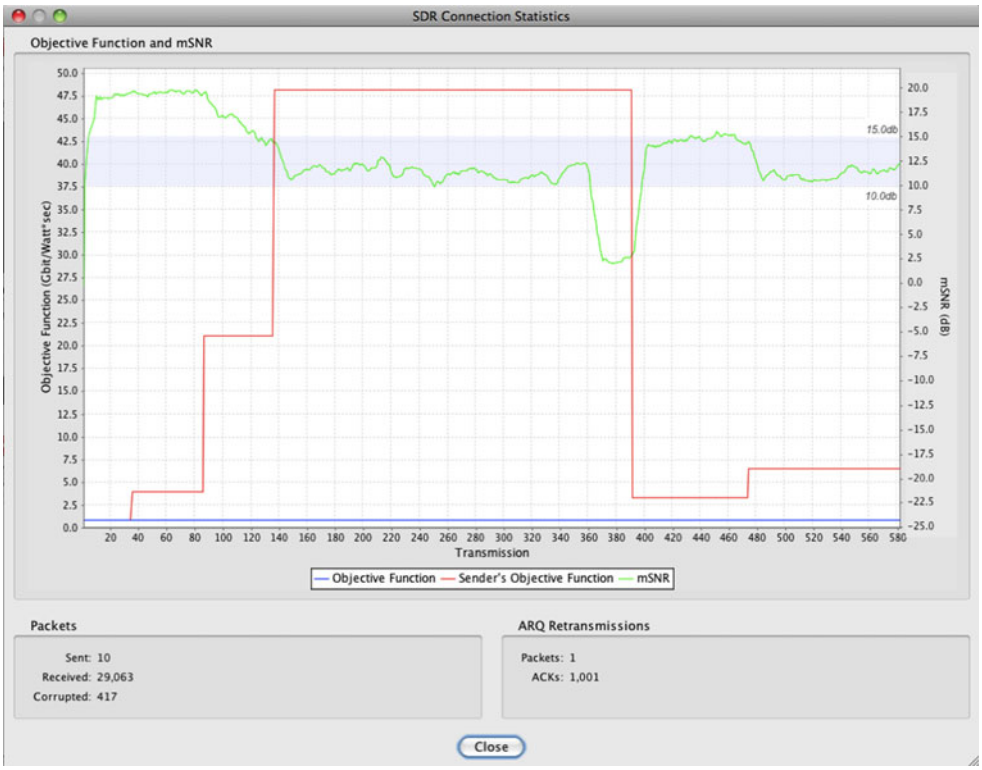

Fig. 11 Policy execution results

this approach versus how much overhead it imposes on the communication link in terms of time delay and spectrum efficiency?

Acknowledgments This work has been partially supported by a DARPA contract through System Planning Corporation (Agreement No. SRA-0775). The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

References

1. Choi, M. J., Hong, W., & Ju, H. T. (2003). XML-based network management for IP networks. *ETRI Journal, 25*, 445–463.
2. Cummings, M., Li, S., & Kokar, M. M. (2008). Activities of SDR forum MLM working group on a language for advanced communication systems applications. In *SDR Forum Technical Conference*, *2008*.
3. Denker, G., Elenius, D., & Wilkins, D. (2009). Cognitive radio technology. In B. A. Fette (Ed.), *Cognitive radio policy language and policy engine* (Chapter 17). Burlington: Elsevier.
4. Endsley, M., & Garland, D. (2000). *Situation awareness, analysis and measurement*. Mahway, NJ: Lawrence Erlbaum Associates, Publishers.
5. Kokar, M. M., & Lechowicz, L. (2009). Language issues for cognitive radio. *Proceedings of the IEEE, 97*(4), 689–707.
6. Kokar, M. M., Baclawski, K., & Eracar, Y. A. (1999). Control theory-based foundations of self-controlling software. *IEEE Intell Syst, 14*, 37–45.
7. Kokar, M. M., Baclawski, K., & Eracar, Y. (1999). Control theory-based foundations of self-controlling software. *IEEE Intell Syst, 14*, 37–45.
8. Kokar, M. M., Hillman, D., & Li, S. (2008). *Towards a unified policy language for future communication networks: A process*. Chicago, IL: DySPAN Conference.
9. Kokar, M. M., Brady, D., & Baclawski, K. (2009). Cognitive radio technology. In B. A. Fette (Ed.), *The role of ontologies in cognitive radios* (Chapter 13). Burlington: Elsevier.
10. Li, S., Kokar, M. M., & Brady, D. Developing an ontology for the cognitive radio: Issues and decisions. In *SDR Forum Technical Conference*, *2009*.
11. Li, S., Kokar, M. M., & Moskal, J. (2008). Policy-driven ontology-based radio: A public safety use case. *SDR Forum Technical Conference, 2008.*
12. Masolo, C., Borgo, S., & Gangemi, A. (2003). *DOLCE: A descriptive ontology for linguistic and cognitive engineering.*. Rome: Technical report, Institute of Cognitive Science and Technology, Italian National Research Council.
13. Mitola, J, III. (2000). Cognitive radio: An integrated agent architecture for software defined radio. *Doctoral Dissertation*, KTH Stockholm, Sweden.
14. Mitola, J., III., & Maguire, G. Q., Jr. (1999). Cognitive radio: Making software radios more personal. *IEEE Personal Commun, 6*, 13.
15. Moskal, J., Kokar, M. M., & Li, S. (2010) Interfacing a reasoner with an sdr using a thin, generic API: A GNU radio example. *SDR Forum Technical Conference*, 2010.
16. Shin, D., & Shim, C. (2005). XNMP—an XML based network management protocol over VoIP. In *Proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Paralled/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks,* Virginia Tech.

17. Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data and Knowledge Engineering, 25*, 161–197.
18. Wang, J., Kokar, M. M., Baclawski, K., & Brady, D. (2004) Achieving self-awareness of SDR nodes through ontology-based reasoning and reflection. In *Software Defined Radio Technical Conference*, *2004*.
19. Wireless Innovation Forum Cognitive Radio Working Group. (2008). *Cognitive radio definition and nomenclature*. Working Document SDRF-06-P-0009-V0.5.0. 30, May 2008.
20. Wireless Innovation Forum MLM Working Group. (2008). *Use cases for MLM language in modern wireless networks*. Working Document SDRF-08-P-0009-V.1.0.0. Retrieved from http://groups.winnforum.org/d/do/1562. Accessed Jan 2009.
21. Wireless Innovation Forum MLM Working Group. (2010). *Description of cognitive radio ontology* (vol. 1.0), WINNF-10-S-0007. Retrieve from http://groups.winnforum.org/d/do/3370. Accessed Sept 2010.

Shujun Li is currently a Ph.D candidate in Computer Engineering at Northeastern University. She received her M.Sc in Signal Processing and Communications from Edinburgh University (2005) and B.Eng in Telecommunications Engineering from Beijing University of Posts and Telecommunications (2004). She is an active researcher in Cognitive Radio, Policy-based Radio Control, Semantic Web and Knowledge Representation. Since 2007, she has served as a major contributor at the cognitive radio standardization groups at IEEE and Wireless Innovation forum. She authored and co-authored over 14 publications in the applications of semantic technologies in cognitive radio and wireless communications. She won the best paper award at Wireless Innovation Forum in 2010 and her book "Flexible Adaptation in Cognitive Radio" will be published in 2012.

Jakub Moskal received his Ph.D in Computer Engineering at Northeastern University (2011) and M.S in Computer Science at Wroclaw University of Technology (2005). He is a Research Scientist with VIStology, Inc. His areas of expertise include ontology engineering, rule-based reasoning, semantic web services, knowledge-based methods in Cognitive Radio, software engineering and object-oriented programming. He has authored or co-authored several publications primarily regarding the Cognitive Radio and rule-based reasoning. He has been a reviewer and a member of the Technical Program Committee at the Software Defined Radio Forum. While pursing his Ph.D. at the Northeastern University he developed a framework that facilitates use of an inference engine at the heart of a Cognitive Radio. This research included development of a universal API between the inference engine and self-controlling software, application of ontology matching and agent-based communication protocols.

Mieczyslaw M. Kokar received his M.S. (1969) in computer engineering and Ph.D. (1973) in computer systems engineering from Wroclaw University of Technology, Poland. Prior to arriving to Northeastern (1984) he was Associate Professor at Millersville University of Pennsylvania (1982–1984), software engineer at HMW Enterprises (1981–1982) and Assistant Professor at the Technical University of Wroclaw (1973–1981). He is an active researcher in Information Fusion, Ontologies, Semantic Web, Cognitive Radios, Self-Controlling Software and Modeling Languages. He has authored and co-authored over 150 journal and conference papers. Kokar has taught various graduate courses in software engineering, formal methods and artificial intelligence. He is on Editorial Board of Journal of Information Fusion and Program Committee member of numerous conferences. He is a senior member of IEEE and member of ACM.

David Brady is currently a Associate Professor in tne Electrical and Computer Engineering department, Northeastern University. He received his M.S.E.E from California Institute of Technology (1983) and Ph.D.E.E from Princeton University (1990). His areas of research interest include, statistical inference in Communication systems, statistical inference in sensing networks and statistical inference in Bioinformatics. He has guided many Doctoral students.