POLICY-DRIVEN ONTOLOGY-BASED RADIO: A PUBLIC SAFETY USE CASE

Shujun Li, Mieczyslaw M. Kokar and Jakub Moskal
Department Electrical and Computer Engineering, Northeastern University, Boston, MA, USA
shli@ece.neu.edu, mkokar@neu.edu, jmoskal@ece.neu.edu

ABSTRACT

Cognitive radios can lead to more reconfigurable and heterogeneous systems due to their flexibility, spectrum efficiency and interoperability. Two main limitations of the current radio systems are: (1) the prescribed, inflexible control structure in the radios; (2) the lack of understanding of their own structure by the radios. Hence, it is impossible to query the capabilities and current state of other nodes and modify their functioning in real-time. In this paper, we show that Ontology-Based Radio driven by policy can be useful in scenarios where dynamic change of radio behavior is required, e.g. dynamic network extension for coverage and reach-back. The combination of ontology and policy provides a more flexible control mechanism in which a program segment is invoked whenever the condition is satisfied. Any modification of the control structure can be accomplished by simply changing the policy.

1. INTRODUCTION

Software Defined Radio (SDR) benefits from its ability to provide versatile intelligent behaviors. However, the change of radio behavior in an SDR requires a time-consuming and inflexible process of software re-deployment, which may take even longer than hardware deployment. In addition, due to the coupling of enforcement and optimization, changes to a component may end up with changes to the entire system. Since imperative programming languages such as C and C++ are typically used in the current radio systems, the control structure of traditional imperative programs is prescribed by the designer at the design time and has a fixed order of invocation, resulting in an inflexible control structure [1].

In this paper, we introduce a policy-driven Ontology-Based Radio (OBR). OBR uses the combination of ontology, policy and policy reasoning to provide the flexibility and interoperability of the communication nodes. Ontology is a mechanism for capturing capabilities, configuration and system state of the radio. Policies are sets of rules about how to change the behavior of the network. A policy reasoner is a component capable of deductive reasoning over the ontology and the rules.

OBR has the following features. First, the operation of OBR is controlled by some policy rather than device-specific software embedded into hardware, i.e. we can define and change the radio operation by changing the policy during its operation. Second, the definition, loading and enforcement of the policies are separated from the implementation and optimization. This decoupling shold simplify the certification and accreditation process, i.e. the policy reasoner and policies only need be certified once and then loaded to any device to change their behavior without additional certification. Also, device and policy can evolve independently [2]. Third, the ontology and policy are expressed in a formal declarative language which can provide extensible standard vocabularies, rule-based inference, and constraint solving capabilities.

In [3], an experimental implementation of OBR was constructed where two radios used ontology-based reasoning to determine the length of the equalizer training sequence. In the experiment, the ontology (written in OWL – Web Ontology Language) was first converted to a Prolog program, which was in turn processed using Kernel Prolog, a Java based Prolog interpreter. The approach presented here is based on BaseVisor [4,5], a Java based *reasoner*, which is a more direct way to implement OBR since we no longer need Prolog as an intermediate step. While the previous experiment is a simple proof-of-concept OBR system, in this paper we show the realization of a real use case [6].

Our main contributions are: (1) We present an OWL ontology to capture and characterize the cognitive capabilities and system state of the network extension use case. (2) We demonstrate the concept of policy-driven radio through reasoning. (3) We show how to organize the policy, the ontology and the reasoner to control the behavior of the radio.

This paper is organized as follows: Section 2 describes the network extension use case in our implementation along with a summary of the functional capabilities. Section 3 presents the OWL ontology we developed for the sole reason of formalizing this use case. In Section 4, we present

the policies which control the radio behavior. In Section 5, we discuss how to implement the policy-driven ontology-based radio for the network extension use case. Conclusions and future work are given in Section 6.

2. THE NETWORK EXTENSION USE CASE

The Network Extension use case comes from the London Bombing scenario which was an actual terrorist event that happened in July, 2005 in London. Bombs exploded on three London Underground trains inside the tunnels. The radios of the first responders in the tunnel did not have connectivity to the above-ground infrastructure. The only means for responders to communicate back to their respective command centers was to run to the nearest station and position themselves at the entrance to the Metro system. It took 15 minutes to walk from the scene to the entrance.

Cognitive radio could be implemented to reconfigure responders' radios to create a peer-to-peer link, which can provide network extension to connect isolated nodes to the infrastructure. With the network extension capability, on-scene responders would have direct communications to command centers without leaving the incident scene or resorting to runners that delayed communications by as much as 15 minutes [6]. The use case diagram is shown in Figure 1.

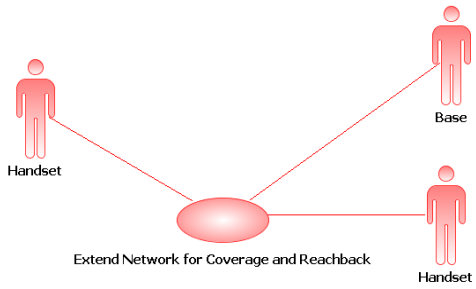

Figure 1: Network Extension Use Case Diagram

The functional capabilities of this use case are summarized as follows:

- Identifying peer radios and determining their connectivity status;
- Authenticating compatible reconfigurable radios;
- Reconfiguring transmission parameters (including frequency, transmission rate, etc.) to form a peer-to-peer link to the infrastructure;
- Adjusting the network topology as responders arrive and depart from the area where coverage is unavailable.

In order to implement the policy-driven OBR for this scenario, we describe the sequence of events as they might occur in this scenario. It is assumed that:

- The radios in this use case and the base stations have the necessary cognitive radio capabilities.
- Since the communication protocol is not the focus here, a simplified version of a proactive ad-hoc routing protocol is used. Each radio maintains a fresh list of destinations and their routes by periodically distributing routing tables throughout the network. In addition, next-hop is assumed to be a single radio.
- Figure 2 shows a typical topology of the network. The circles indicate the communication radius of each radio.
- Radios are preregistered with the base station.

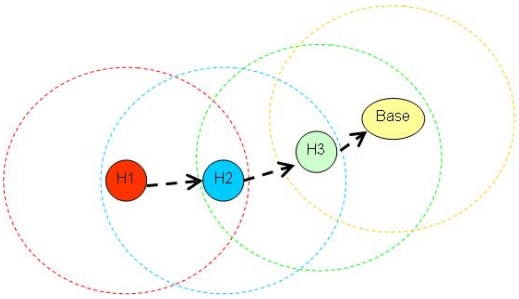

Figure 2: Topology of Network Extension Use Case

The timeline of this use case is shown as follows (see Figure 3):

1. First-Responder's radio is disconnected from the infrastructure
   a) The first-responder's radio (H1) senses the RF environment. The term "senses" means scanning a group of forward control channels in search of the strongest base station signal. The forward control channel is used for transmission of control messages from the base station to the mobile. If none of the control channels has signal strength above a usable level, the radio determines it's disconnected from the base station.
2. First-Responder's radio sets up a peer-to-peer link to the base station
   a) H1 broadcasts a query message to its neighbors for their information. Each radio maintains a route to each other radio in the network. The routing table contains ID (for example: IP address, MAC address, etc), next-hop ID, destination ID, and hop count. Initially, next-hop ID and hop count are set to a reasonable hop limit.
   b) H2, one of H1's neighbors, senses the RF environment and receives the query message from H1. Then it sends back an answer message to H1 and re-broadcasts the same query message to its neighbors. The answer message contains its own ID, spectral information, geographic or relative position and so on.
   c) H1 receives the answer message from H2 and updates its routing table. There are different criteria to choose the next-hop.
   d) H3, one of H2's neighbors, senses the RF environment and receives the query message from H2.

Then it re-broadcasts the query message to its neighbors and sends back an answer message to H2.

e) H2 receives the answer message from H3 and updates its routing table.

f) Repeat (b) to (e) until the sender finds a path to the destination, which is the base station in this case.

g) The base station sends a "route" reply message, traversing back along the desired path to H1.

h) Route discovery is finished when the route reply message is received by H1.

3. First-Responder's radio transmits data packet to the base station

a) H1 sends a command message to the next-hop radio (H2), requesting H2 to be ready to receive data.

b) H2 receives the command message and sends a confirm message to H1 if it's ready to receive data.

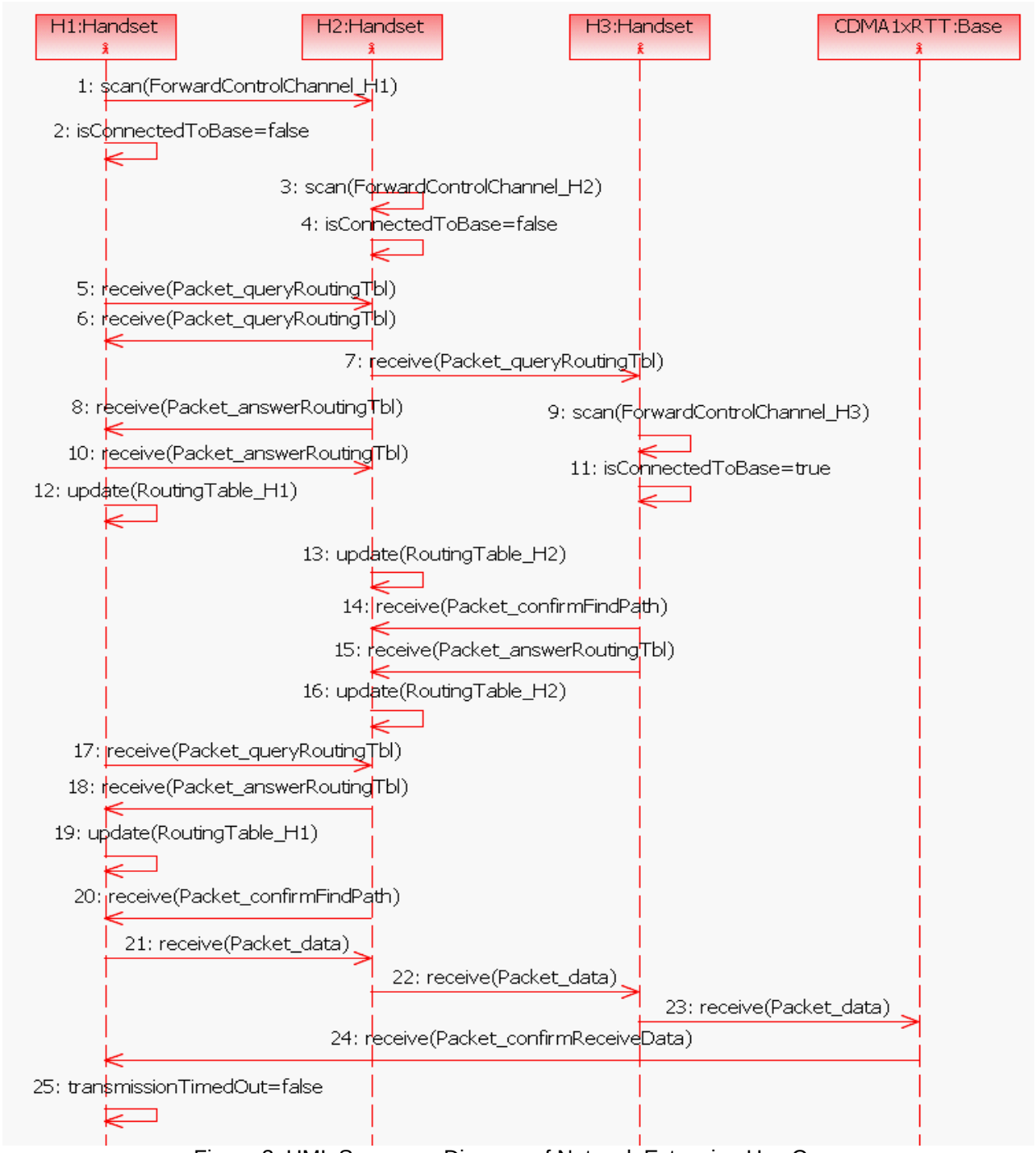

Figure 3: UML Sequence Diagram of Network Extension Use Case

c) H1 sends a data message to H2.

d) H2 receives the data message and forwards the data to its next-hop radio (H3).

e) When the data message arrives at the destination, the destination radio sends back a confirm message (acknowledgement) to H1.

f) If H1 receives the confirm message within a predefined period of time, then the data transmission is finished. Otherwise, the data is considered to be lost; then the system repeats (a) to (e).

The topology of the network is dynamically changing as responders arrive and depart from the area where coverage is unavailable. Therefore, the radios must enable both periodic and event-triggered routing table updates.

The events are also represented in the UML sequence diagram in Figure 3. The boxes at the top of the figure represent radios involved in the use case. The vertical lines represent "life lines" or "time lines" of the radios with the time direction pointing downwards. Interactions between particular radios are shown by horizontal arrows annotated with the message types. All radios must have the knowledge of how to respond to particular types of events. All messages are expressed in the modeling language that all the radios can "speak".

3. OWL ONTOLOGY REPRESENTATION

We developed an ontology for the sole reason of formalizing the network extension use case. This ontology defines the basic classes and properties that were required to implement this use case. A graphical representation of this ontology is shown in Figure. The ontology was formalized in OWL using the Protégé tool [7]. Each rectangle in this figure represents a class. The first row in the rectangle shows the name of the class, followed by several rows, each of which shows a property of that particular class, either a *data type* property or an *object* property. Arrows between classes are annotated by property names. Such arrows represent object properties. Each arrow can be read as a triple *<Class1 propertyName Class2>*, e.g. *<Radio hasComponent RadioComponent>*. An arrow connecting two classes annotated by "*isa*" represents subclass relationship, e.g. *RoutingTable* is a subclass of *RadioComponent.*

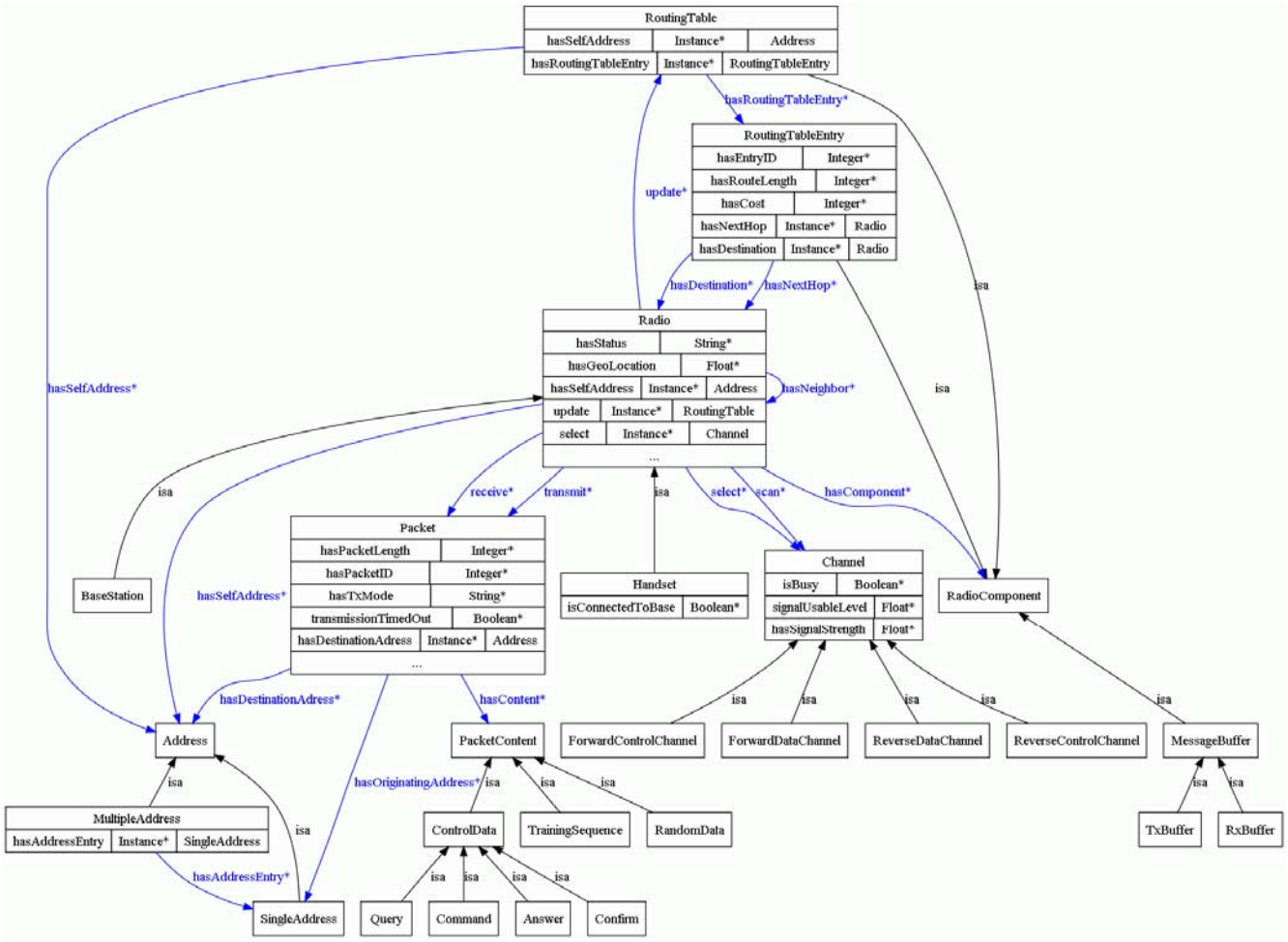

Figure 4: Ontological representation of the Network Extension Use Case. Isa arrows are black. Other arrows are blue.

4. THE RULES

A policy is expressed as a set of rules. The rules are written in the form of if/then statements. The "if" part of the statement is called the "body" part; the "then" part of the statement is called the "head" part. BaseVisor is used as the reasoning engine in our experiment; it implements forward chaining by using a Rete network. In the BaseVisor rule language, both heads and bodies are expressed as triples. The triple-based rules are added to the rule base and then compiled into Rete networks, generating the nodes of the Rete network. Running the Rete network causes the rules to fire and facts to be added to the fact base. A particular rule is triggered when the triple patterns in the body of the rule match the facts found in the fact base. The head of one rule may feed the body of another rule. Hence the behavior can be flexibly controlled by the rules.

The set of rules for the network extension use case includes rules to (1) check the signal strength to determine the connectivity status; (2) check whether the received packet is *MyPacket*; (3) query of neighbor's information when a radio is disconnected from the base; (4) send back an answer message when a radio receives a query from others; (5) process the answer message and update the routing table; (6) send a route reply message traversing back along the desired path to the starting hop when a radio finds a path to the base; (8) store and forward the data packet to the next hop; (9) send back an end-to-end acknowledgement when the data packet arrives at the destination.

In the BaseVisor syntax, the subject, predicate or object element can be a resource, a data type value or a variable. For simplification, we express the rules in a simplified way, i.e. the element with "#" refers to a resource and the element without "#" refers to either a data type value or a variable. For instance, in triple *<FCC rdf:type #ForwardControlChannel>*, *FCC* is a variable; *ForwardControlChannel* refers to a class specified in the OWL ontology; "*rdf:type*" means FCC is an object of *ForwardControlChannel* class. The following are examples of some of the rules used in the network extension use case.

The following is the rule to check the connectivity status. If the signal strength of the forward control channel is below a pre-defined usable level, then it's determined that the radio is disconnected from the base.

```
<rule name="checkConnectivity">
IF
   (FCC, rdf:type, #ForwardControlChannel)
   (#MyRadio, #select, FCC)
   (FCC, #hasSignalStrength, SignalStrength)
   lessThan (SignalStrength, #UsableLevel)
THEN
   (#MyRadio, #isConnectedToBase, false)
```

The following rule says that if a radio is connected to the base and the system status is "idle", then the radio broadcasts a query to its neighbors its routing-table information.

```
<rule name="queryNeighborRoutingTbl">
IF
   (#MyRadio, #isConnectedToBase, false)
   (#MyRadio, #hasStatus, idle)
   (#AllNeighbors, #hasAddressEntry, AN)
THEN
add the followings facts to <AllNeighbors>'s
knowledge base:
   (#MyRadio, #transmit, #Packet_queryRoutingTbl)
   (#Packet_queryRoutingTbl, #hasTxMode, broadcast)
   (#Packet_queryRoutingTbl,
#hasDestinationAddress, AN)
   (#Packet_queryRoutingTbl,
#hasOriginatingAddress, #MyAddress)
```

The following rule says that if Radio1 receives a query of its routing table from Radio2, then Radio1 sends its routing table toRadio2.

```
<rule name="processPacketQueryRoutingTbl1">
IF
   (#MyRadio, #receive, #Packet_queryRoutingTbl)
   (#Packet_queryRoutingTbl, #isMyPacket, true)
   (#Packet_queryRoutingTbl,
#hasOriginatingAddress, YourAddress)
   (YourRadio, rdf:type #Radio)
   (YourRadio, #hasSelfAddress, YourAddress)
   (#MyRadio, #hasComponent, MyRoutingTable)
   (MyRoutingTable, rdf:type, #RoutingTable)
   (MyRoutingTable, #hasRoutingTableEntry,
MyRoutingTableEntry)
   (MyRoutingTableEntry, #hasDestination,
MyDestinationRadio)
   (MyRoutingTableEntry, #hasNextHop,
MyNextHopRadio)

THEN
add the following facts to <YourRadio>'s knowledge
base:
   (#MyRadio, #transmit, #Packet_answerRoutingTbl)
   (#Packet_answerRoutingTbl, #hasTxMode, unicast)
   (#Packet_answerRoutingTbl,
#hasDestinationAddress, YourAddress)
   (#Packet_answerRoutingTbl,
#hasOriginatingAddress, #MyAddress)
   (#MyRadio, #hasComponent, MyRoutingTable)
   (MyRoutingTable, #hasRoutingTableEntry,
MyRoutingTableEntry)
```

```
   (MyRoutingTableEntry, #hasDestination,
MyDestinationRadio)
   (MyRoutingTableEntry, #hasNextHop,
MyNextHopRadio)
```

The following rule says that if a radio is connected to the base and there is a data packet in the transmitting buffer, then the radio looks up its routing table and sends the data packet to the next hop.

```
<rule name="transmitRandomData">
IF
   (#MyRadio, #isConnectedToBase, true)
   (#MyRxBuffer, #isEmpty, false)
   (#MyRoutingTable, #hasRoutingTableEntry,
MyRoutingTableEntry)
   (MyRoutingTableEntry, #hasDestination,
#Address_CDMA1xRTT)
   (MyRoutingTableEntry, #hasNextHop,
MyNextHopRadio)
   (MyNextHopRadio, #hasSelfAddress,
NextHopAddress)

THEN
add the following facts to <MyNextHopRadio>'s
knowledge base:
   (#MyRadio, #transmit, #Packet_data)
   (#Packet_data, #hasDestinationAddress,
NextHopAddress)
   (#Packet_data, #hasOriginatingAddress,
#MyAddress)
   (#Packet_data, #hasStartingHopAddress,
#MyAddress)
   (#Packet_data, #forwardFlag, true)
```

In our experiment, we implemented 13 rules for the Network Extension use case.

5. IMPLEMENTATION

In our experiment, two-way communication between 4 radios is implemented using an emulated channel. Each radio has its own Reasoning Component (RC). The input of the RC contains static facts, dynamic facts and policies, as shown in Figure 5. Static facts (usually referred to as "T Box") are the basic terms in communications domain, usually including classes and properties. Dynamic facts ("A Box") are the facts only available as the radio is operating. Since the ontology already defines the basic concepts, dynamic facts are usually the instances generated by the system at the run-time. Policies ("R Box") are the rules specified in declarative form, which describe how to react to particular situations. In our implementation, the 4 radios share the same T box while each of them has its own individual R box and A box.

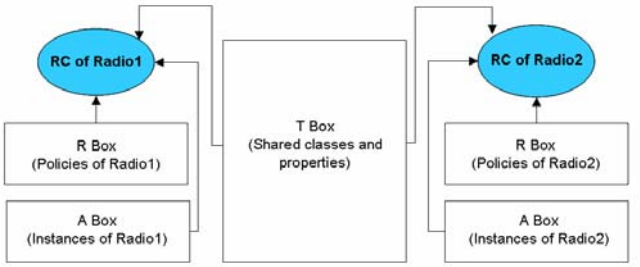

Figure 5: Implementation Demonstration

The following shows an example trace of our simulation. Note that H1, H2 and H3 are handsets; CDMA1XRTT is the base station.

```
[1] H1 transmits Packet_queryRoutingTbl to
rad:Address_H2
[2] H2 receives rad:Packet_queryRoutingTbl from
rad:Address_H1
[2] H2 transmits Packet_answerRoutingTbl to
rad:Address_H1
[2] H2 transmits Packet_queryRoutingTbl to
rad:Address_H1
[2] H2 transmits Packet_queryRoutingTbl to
rad:Address_H3
[1] H1 receives rad:Packet_queryRoutingTbl from
rad:Address_H2
[1] H1 receives rad:Packet_answerRoutingTbl from
rad:Address_H2
[1] H1 transmits Packet_answerRoutingTbl to
rad:Address_H2
[3] H3 receives rad:Packet_queryRoutingTbl from
rad:Address_H2
[3] H3 transmits Packet_confirmFindPath to
rad:Address_H2
[3] H3 transmits Packet_answerRoutingTbl to
rad:Address_H2
[2] H2 receives rad:Packet_answerRoutingTbl from
rad:Address_H1
[2] H2 receives rad:Packet_answerRoutingTbl from
rad:Address_H3
[2] H2 receives rad:Packet_confirmFindPath from
rad:Address_H3
[2] H2 transmits Packet_confirmFindPath to
rad:Address_H1
[1] H1 receives rad:Packet_confirmFindPath from
rad:Address_H2
[1] H1 transmits Packet_data to rad:Address_H2
[2] H2 forwards Packet_data to rad:Address_H3
```

```
[2] H2 receives rad:Packet_confirmFindPath from
rad:Address_H1
[3] H3 forwards Packet_data to
rad:Address_CDMA1XRTT
[4] CDMA1XRTT receives rad:Packet_data from
rad:Address_H3
[4] CDMA1XRTT transmits Packet_confirmReceiveData
to rad:Address_H1
[1] H1 receives rad:Packet_confirmReceiveData from
rad:Address_CDMA1XRTT
[3] Terminated.
[2] Terminated.
[4] Terminated.
[1] Terminated.
```

Initially, H1 and H2 are disconnected from the base; H3 is connected to the base. Hence, H1 and H2 trigger the *queryNeighborRoutingTbl* rule and then keep transmitting a query of their neighbors' routing information until they find H3. Then, a "path-confirm" message is sent from H3 to H2 and H1. After the peer-to-peer link is set up, H1 triggers the *transmitRandomData* rule and starts to transmit a data packet through the path. The mid hops forward the data packet until the data arrives at the end hop – CDMA1xRTT. Finally, the end hop sends an end-to-end acknowledgement to the starting hop – H1. The result verified the logic correctness of the rules and showed that the behavior can be flexibly controlled by the rules. The rules can be modified or more rules can be added without any modification of the underlying structure of the communication protocols.

6. CONCLUSIONS AND FUTURE WORK

We have presented a mechanism in which the behavior of Ontology-Based Radio is driven by policy. We have implemented the network extension use case – a public safety domain use case - in which three handsets and one base station are automatically reconfigured to create peer-to-peer links for reachback and coverage. In order to complete the implementation of this use case, we had to develop an OWL ontology to define the language for representing both the capabilities of the communication nodes and the messages to be exchanged among the communicating nodes. Moreover, to demonstrate the cognitive capabilities on the example of the network extension use case, we had to develop a set of rules to both implement the functional capabilities of the radios and to control the behaviors of the nodes. The implementation shows how to organize the policies, the ontology and the reasoner so as to control the behavior of the radio.

At this time we are not aware of this kind of implementation of other use cases relevant to public safety and commercial applications. It should be noted that the OWL ontology presented in this paper is not applicable for other use cases, e.g. Dynamic Spectrum Access. In the future, we will continue our work on building a more comprehensive ontology which can characterize the capabilities of the physical layer and the data link layer. Moreover, we will work on the development and the structuring of a comprehensive set of policy rules.

7. REFERENCES

[1] B. A. Fette (Ed.), Cognitive Radio Technology, Newnes Publication, August, 2006.

[2] D. Wilkins, G. Denker, M.-O. Stehr, D. Elenius, R. Senanayake and C. Talcott, "Policy-Based Cognitive Radios", IEEE Wireless Communications, 14(4), pp41-46, August. 2007.

[3] J. Wang, M.M. Kokar, K. Baclawski, and D. Brady, "Achieving Self-Awareness of SDR Nodes through Ontology-Based Reasoning and Reflection", Software Defined Radio Technical Conference SDR'04.

[4] C. Matheus, K. Baclawski and M. Kokar. BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML and R-Entailment Rules. In Proceedings of the 2nd International Conference on Rules and Rule Languages for the Semantic Web, Athens, GA, Nov. 2006.

[5] BaseVISor, www.vistology.com/BaseVISor.

[6] Public Safety Special Interest Group, "Use Cases for Cognitive Applications in Public Safety Communications Systems, Volume 1: Review of the 7 July Bombing of the London Underground", Working Document SDRF-07-P-0019-V0.0.3, Version 1.0.0, September 18, 2007.

[7] Protégé, http://protege.stanford.edu.