A Component-based Architecture for Cognitive Radio Resource Management

Mohamed Ahmed*, Vinay Kolar†, Marina Petrova†, Petri Mähönen† and Stephen Hailes* *Department of Computer Science, University College London, UK.
email: {m.ahmed and s.hailes}@cs.ucl.ac.uk †Department of Wireless Networks, RWTH Aachen University, Germany.
email: {vko, mpe and pma}@mobnets.rwth-aachen.de

***Abstract*—Cognitive Radio Networks are envisioned to solve the problem of spectral scarcity in wireless networks; through providing highly configurable radios and protocol stacks to support the application of a variety of efficient and possibly cross-layered solutions. However, the large numbers of hardware and software modules involved in realising these goals raises a fundamental design problem. Specifically, how do we do construct scalable and extensible systems to work across heterogeneous systems and help realise their full potential.**

To address these issues, we propose a component-based approach to the construction of the control and management software for radios. We propose generic interfaces to support heterogeneity and portability. Our architecture supports dynamic policy updates and enforcement through a Policy Engine. Finally, we show the realisation of the proposed implementation architecture through a system example.

I. Introduction

The increase in the numbers of wireless devices and applications has led to a great demand for the sparse capacity of wireless networks, especially in the license-free ISM bands. Cognitive Radios (CR) and Dynamic Spectrum Access (DSA) provide a promising avenues of research to solve the above problem [9], [1]. Unlike classical radio devices, the CR paradigm emphasises the reconfiguration of radio functionality at a fine-granularity through permitting access to a wide range of spectrum and choice of custom modulation mechanisms that can be exploited to achieving greater spectrum efficiency.

Several factors have contributed to the growth of research in this area. Radio manufacturers have started to create flexible software-defined radios that reveal the low-level radio parameters and functionalities, and support the dynamic reconfiguration of the complete protocol stack [14], [8]. As a result, protocol and application designers have been exploiting this rich set of functionalities to design novel mechanisms that enable cross-layer and cross-technology solutions [1].

However, in aiming to realise the cognitive radio objective, we face a number of challenges. Amongst them, first, even though various device manufacturers expose different sets of radio functionalities most use proprietary interfaces. Second, custom optimisation modules and protocols exploit the configurable radio parameters though utilising the input from several layers and tweaking the system parameters using a non-standard approaches. Together, these issues make it hard to integrate different modules in a single system, and make it difficult to attain Mitola's original vision [9] of building holistic systems that are aware of their state and environment, and capable of adjusting their behaviour in response to changes.

Though there exits several framework proposals for Cognitive Radio Management(CRM), for example [12], [13], too often they are too narrowly focused on the optimisation task, and lack a holistic perspective on the problem. To address this issue we must consider the problems faced when designing resource management solutions for cognitive radios and how they impact on the design task.

The primary issue here concerns the factors to consider in the general resource management task, e.g. protocol stack, hardware, deployment environment and etc. Resource management frameworks may be expected to gather information from large variety of sources, and provide methods to perform actions upon them. Given the problem at hand, i.e. latency sensitive applications, they may be expected to support real-time operation. The heterogeneity of candidate deployment environment means (hardware, OS, and etc.) raise the issues of portability and extensibility. The complexity of specifying cross-layer optimisation tasks, especially as the number objectives increases means that we must consider how we specify individual tasks and how we address their coordination.

This work address some of the concerns raised for Cognitive Resource Management architectures and proposes a component-based framework that supports; (i) generic interfaces to support heterogeneity and portability, (ii) policy management to handle stakeholder and user policies, and (iii) a behaviour abstraction to support extensibility and modularise the definition of individual optimisation/control goals and their coordination. We present an overview of our prototype and refer the reader to [12] for a discussion of the generic CRM framework.

II. Problem Statement

Realising and exploiting the capabilities offered by a cognitive radio network requires a CRM framework that manages the various hardware and software modules and building a single monolithic resource management unit leads easily to unscalable systems, that are difficult to extend and port. Because cognitive radio terminal may consist of several modules that attempt to pursue individual objectives in either a cooperative

or a competitive manner, the CRM should provide methods to coordinate their action and resolve the conflicts that may arise. To address the concerns discussed we define the following requirements when designing resource managers for cognitive radio systems:

- *Extensibility*: To enable systems designer to work incrementally and experiment with different approaches to a problem, the CRM must be modular and extensible. The CRM must as much as possible separate the manner it is constructed from the problems it addresses. In practise, this means support for well defined interfaces, the dynamic binding of code and conflict resolution to resolve binding errors.
- *Portability*: Given the heterogeneity of radio hardware and target deployment environments, a natural goal for the CRM is to increase the portability and application of the optimisation and control routines it provides. The CRM must therefore provide consistent transparent interfaces to access both hardware and the network stack.
- *Policy-awareness*: Because spectrum license-owners, regulatory bodies, and users may restrict spectrum access through the definition of transmission policies [5], the CRM must specify methods for representing, reasoning about and enforcing these policies.
- *Support for distributed operation*: The CRM must be capable of managing resources that may be distributed over a network, since features such as control channel negotiation are often required to coordinate the behaviour of terminals. Further, to support the portability requirement, this requirement should also be transparent to the mechanisms that depend on it.
- *Performance*: To support possible low-latency applications like transmit-rate control, the CRM must provide support for real-time operation.
- *Complexity reduction*: Given the complexity of specifying and implementing cross-layer optimisation strategies, the CRM must provide mechanism to specify them in modular manner and facilitate their incremental extension and the coordinate between them.

In summary, our requirements stress support for extensibility, generalisation, and experimentation.

III. Related Work

Though there exist several cognitive radio architectures and frameworks in the literature, many proposals address only the DSA problem [20]. Others are still high-level and only limited real implementations demonstrate a small set of cognitive features.

The most relevant work to our CRM design is the concept of Cognitive Engine (CE) [15]. The engine is capable of learning the behaviour of the radio in different environments over time, and intelligently adapts the communication stack to fit the new wireless communications scenarios based on a set of objectives and constraints. However, so far the optimisation of the communication stack has been constrained to solve a subset of the CR problems: setting up PHY layer parameters (modulation, transmit power, number of sub-carriers, etc.) using genetic algorithms.

A component based Reconfigurable Node design was introduced by Trinity College Dublin for reconfiguration and observation of the radio and network stack [17]. As a result of a joint effort between Virginia Tech and Trinity College, a framework for implementation of cognitive functionalities that links the concepts of CE and Reconfigurable Node has been proposed in [11]. The main underlying idea of the framework is to facilitate network stack composition and management. Still, no real implementation solution for the controlling structure is given.

Moreover, there are number of key architectural issues that are not addressed. For example, the framework does not define specific interfaces to enable flow of context information from the protocol stack to the cognitive engine, and feed the optimised parameters back. The work does not address the problem of scheduling and managing different control actions, or the capability of distributed system support, and etc.

IV. System Example

In order to assist the understanding of the architecture presented in Section V, we present an example cross-layer application that aims to jointly optimise video and MAC parameters in order to maximise the video throughput. This early prototype has been implemented as a part of the ARAGORN [16] project so as to test architectural primitives and enables us to refine our model.

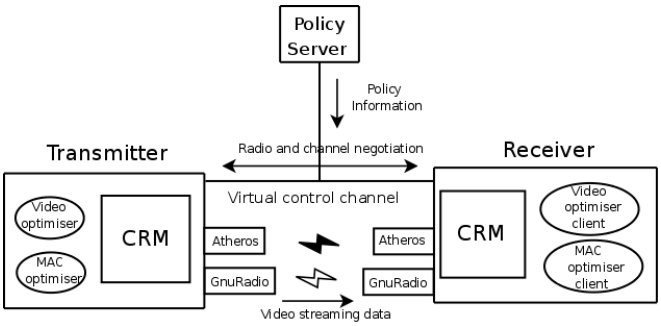

Fig. 1. A high-level overview of our Video-MAC cross-layer optimisation tested. In this setup, the transmitter streams a video to the receiver, which in return sends back channel measurements and statistics concerning the quality of service. Based on this information, the transmitter evaluates its performance and adapts its video and MAC parameters to maximise the performance measure.

Figure 1 illustrates a system setup, whereby a video transmitter, receiver and policy server nodes are present and the wireless nodes are equipped with multiple interfaces (Atheros and GnuRadio).

Although the scenario given presents a large number of input parameters and candidate actions that can be exploited, to simplify our initial experimentation we only look at a small subset. From the radio, we are interested in how we can use the SNR, jitter and contention information across different channels to alter the performance of the MAC by modifying the transmission power, transmit-rate and channel selection.

For video streaming we use the Video Conferencing tool (VIC) [10] with the H.261 codec. Unlike MPEG4 codec, H.261 provides analog control over the source rate of the video transmission, and as a result gives our application continuous control over the video transmission rate. To asses the video performance, we take as inputs the percentage of missing packets, frames and the video throughput (over a given interval). For actions we modify the frame-rate and quantisation at the sender, thus altering the source rate of the video traffic. Although not presented in this example, our architecture supports advanced QoS based policy constraints for video application and the system can easily be generalised to a multi-hop network with multiple nodes.

The aim of the optimisation task is maximise the video throughput in the presence of external noise (we introduce an external noise source to generate the disturbance). The optimisation task is performed through applying reinforcement learning [19]. Reinforcement learning provides a simple method to specify the optimisation task, through linking the actions that can be performed in each of the states of the system to rewards that reflects the goodness of the action, with respect to the state. To perform on-line learning without the large state-space problems associated with common methods, i.e. Q-learning [2], the learning task is formulated as a function approximation, with a feed-forward neural network providing the generalisable function approximator [18]. The inputs to the neural network are the sensory reading we collect from the environment, and the action sets available, and the outputs represent the action selection probabilities.

V. System Architecture

To support the requirements listed in Section II, we have make use of a component-based approach. Component-based approaches are typically proposed to simplify the design and management of complex software through the modularisation of individual constituent parts [6]. As depicted in Figure 2, the proposed architecture consists of: 1) CRM core that is composed of a set of a pluggable behaviour components, 2) generic interfaces components to support transparent access to the underlying system 3) distributed control and co-ordination modules, and finally 4) a policy engine.

In this context, components are encapsulated units of functionality and deployment. Our components model utilises the RUNES component model [6] and components only may only interact through well defined interfaces and receptacles, as shown in Figure 3. Complex and modular systems may be implemented through the specification of specific units of functionality (components) and their interactions (interface/receptacle bindings). The model also defines *Capsules*, *Loaders* and *Binders*. Capsules group instances of components to form a single instance of a system. While loaders and binders enable a capsule instance to dynamically manage its constituent components - through the loading, unloading and the bindings of the components.

It is worth stressing at this point that because of the component focus of our architecture, all parts of the CRM may be implemented as components utilising well defined interfaces, and consequently enabling developers to abstract away from the specific requirements of possible target deployment environments.

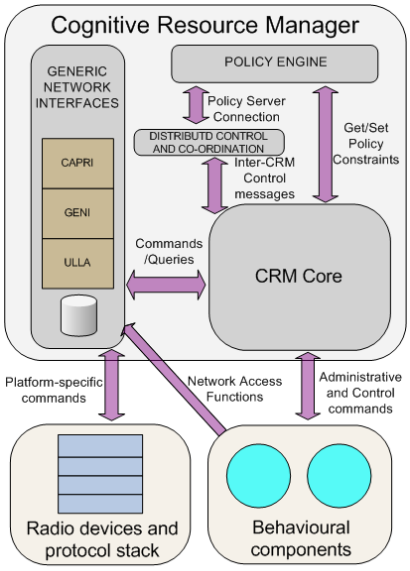

Fig. 2. The high-level architecture of the ARAGORN Cognitive Resource Manager (CRM).

A. CRM core

The CRM core is essentially a capsule and plays two vital roles in our system. First, it provides an the administrative framework to coordinate the construction of the system, and second it provides the control primitives used to coordinate the actions of the components. The following sub-sections discuss the abstractions provided by these roles in more detail and highlight how they may be used to satisfy the constraints listed in Section II.

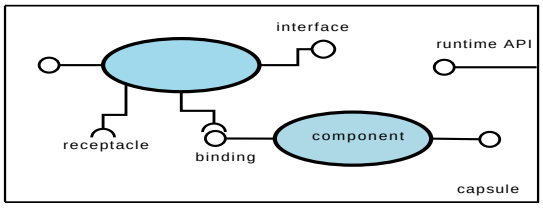

Fig. 3. The RUNES component model [6] defines the notions of *capsules*, *caplets*, *loaders* and *binders*, all of which may be implemented as components. Components interact with each other through well defined interfaces and receptacles with intern list, the functions that they offer and the dependencies between them.

1) Administrative framework: So that we may construct systems in a modular and dynamic manner, e.g. from individual components that are bound at run-time, we must provide the methods and mechanism that monitor the state of the system, resolve conflicts as and when they arise and ensure that the appropriate operating system resources are provided. These administrative functionality are provided by a *component manager* module within the CRM core.

We restrict each CRM core to contain only one component manager, this way, it aware of all the functions the CRM provides, and can play a coordinating role - for example providing resource allocation and dependency resolution. Further, having the component manager means that the system administrator has a holistic overview of the CRM, and can perform tasks such as profiling individual components.

2) Control framework: The control framework of the CRM is defined to address the concerns of the different control and optimisation aspects that may loaded in the CRM, and as such, it provides the abstractions for dealing with decision making aspects of the system. The control framework is composed of; *behaviours*, *generic interfaces*, *action brokers*, *action resolvers distributed control components* and a *policy manager*.

B. Behaviours

We use the abstraction of behaviours to refer to the individual tasks that may be pursued by the CRM. With this abstraction, a behaviour is simply a unit of goal directed functionality and as such, may be mapped to a single component or a number of tightly bound components. An individual behaviour are defined in terms of a data-flow model, as shown in Figure 4. For example, the source-rate adaptation task of the video in Section IV is represented as individual behaviour that is independent in implementation and specification from the rest of the system. This way, complex system may also be decomposed into smaller more manageable units.

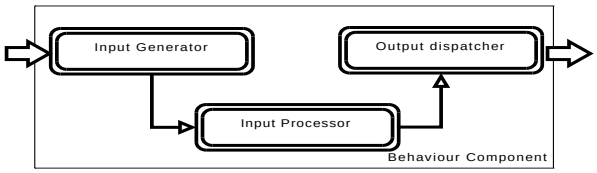

Fig. 4. Data Flow Model of *Behaviour* components: the input generator is responsible for isolating the component from data sources. The input process is responsible for performing the evaluation, and the output dispatcher is responsible for communicating any decisions made.

Behaviours are implemented as components, and in order to interface with the rest of the system, they must provide handles (in the form of interfaces and receptacles) both to interact with the other components, e.g. the component manager their associated action broker and other elements of the system that rely on directly.

From a control perspective, a behaviour has the following properties: first, it collects some input for processing. Second, it evaluates the input(s) collected and last suggests some action(s) to perform based on the evaluation. Therefore, As depicted in Figure 4, a behaviour is simply a coupling of some set of input, processing, and output mechanisms.

To support this coupling, our architecture make uses these distinctions as primitives and defines; input generators, input processors, and output dispatchers. The input generator is responsible for providing generic interfaces to collecting and formatting input for processing - in practise, this is a binding to an interface monitor, such as the generic interfaces that are described later in Section V-C, or a set of constraints and goals in the case of a policy enforcing behaviour. The input processor is responsible for evaluating the input and performing the decision making task, which should ideally lead to a local decision. Finally, the output dispatcher is responsible for communicating the decision either for actuation or further processing.

C. Generic Interfaces

Generic interfaces abstract the varied and complex implementation of various protocol layers and present them in a consistent and systematic manner, thereby providing portability and supporting heterogeneity across different radio devices and operating platforms.

We abstract the radio and link level interfaces through utilising *ULLA* (Universal Link Layer API) [3]. A similar ARAGORN project developed interface, *GENI* (GEneric Network Interface), enables the detailed monitoring and configuration of the transport and network layers. Together, ULLA and GENI provide interface functionalities through the provision of a generic and portable API.

We use Common APplication Requirement Interface (*CAPRI*) to interface between applications and the CRM, such that application are able to define their goals and objective in a utility compatible (quantifiable) manner. In effect, CAPRI provides the utility function based descriptions for *goals*, *payoffs*, and *restrictions*. CAPRI is one of the most challenging parts to implement as a part of ARAGORN architecture [16].

ULLA and GENI interfaces enable querying and altering various networking parameters using a query language similar to SQL. While CAPRI uses an advanced language to query and specify the application objectives (e.g. maximise throughput or minimise delay functions).

D. Action brokers

The definition of behaviours as distinct and individual operational units raises the concern of coordinating their function in order to steer the system towards a system objective/goal. Conflicts can easily arise in the resource management task, since several behaviour modules co-exist and aim to independently optimise their tasks. To address this concern we define the abstraction of action brokers; whose job it is to collect together sets of dependent behaviours and provide a means to choose between their proposed actions - in effect the behaviours dispatch their decisions to action brokers. In this context, two given behaviours may be defined to be independent of each other only if the execution of their suggested actions does not raise a conflict. Consequently, a given behaviour may only appear only in one action broker, and a CRM may host numerous action brokers.

Furthermore, for finer grained control, this abstraction enables us to define action brokers recursively, thereby simplifying the grouping of dependent behaviours and allowing for a greater flexibility in the method we may use to choose between their actions, i.e given as set of dependent behaviours, we may subdivide them into a sub-group that resolves via a prioritisation policy and a set that operates in a competitive manner.

Relating back to our example, the action broker is the mechanism that captures that the MAC and video adaptation are working on the same problem - through defining that their actions are not independent from each other. Alternatively, if we are interested in adapting the MAC and video independently of each other, then we would place them under different action brokers.

E. Action resolver

Resolvers are contained within the action brokers and implement the resolution method to be used by the broker, in effect action resolvers define how a broker chooses between the candidate actions presented by the behaviours it groups.

While the specification of a resolution algorithm depends on the problem at hand, there are two broad methods of decomposition, available; horizontal and vertical (see Figure 5). Under horizontal decomposition, the action resolution mechanism utilises either a *cooperative* (weighted combinations) or a *competitive* (argmax/argmin) operator to select the output action. With vertical decomposition, the resolver may group the behaviours in a hierarchical order based on prioritisation, for example using subsumption based controllers derived from understanding how the outputs of behaviours override one another, or it may apply an arbitrary policy. In this way, the final output selected by the action broker is the result of applying the resolution routines given in the action resolver

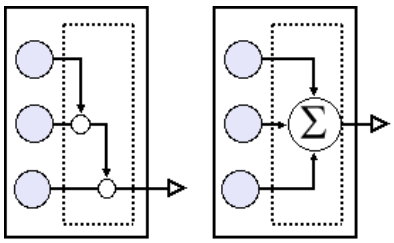

Fig. 5. The CRM behaviours coordination abstraction uses action brokers (solid line) to contain dependent behaviours (filled circles) and the action resolution mechanisms (dotted line) to select between their suggestions. In practise, behaviours report their suggested actions to the resolver which applies its resolution algorithm(s), for example, a prioritisation hierarchy or a weighted sum to select the individual action to execute.

The advantage offered by the action broker/resolver abstraction is that it systematises the process of defining complex control modules through decomposition. In this way, more complex system than our prototype may be easily modularised, enabling designers to experiment with different combinations of optimisation and control pairings in a systematic manner.

For a practical example, if we look back at our example in SectionIV, we notice that there are number of ways to break up the optimisation and control task, i.e. a hierarchical decomposition of the video and MAC, whereby the video output overrides the MAC output, or a weighted combination the outputs of both the controllers. However, the problem here is that without any extra knowledge, it is difficult to gauge the advantages offered by the different techniques a priori. Our modular approach means that we are able to experiment with different control methods in a systematic manner.

F. Distributed control and co-ordination

The CRM must provide mechanisms to support distributed control and co-ordination in a transparent manner and as such, distributed control and co-ordination components provide the mechanism and abstractions to support this operation.

The main behaviour that makes use of this component is the control channel negotiation and utilisation. Virtual or real control channels may be negotiated by CRMs of the nodes using different mechanisms. For example, a static and explicit control channel or a dynamic on-demand control channel can be used. We do not discuss specific approaches to negotiating control channel in this paper, however, the existence of common control channel is a quite clear requirement for distributed and cooperative CRM architecture.

G. Policy Engine

Policies are required to provide operation constraints to the CRM, and they may be static or dynamic with respect to time and geographical location, and this choice has an influence on the manner in which they are handled. The Policy Engine ensures the remote download of the applicable policies from appropriate policy server and applies the constraints they list to the CRM control framework.

ARAGORN policies are mostly specified as constraints using a declarative language like CoRaL [7]. We are currently pursuing advanced policy specification through a more powerful interface that is similar to CAPRI.

The ARAGORN CRM architecture maps well to out prototype application; componentisation offers us a modular and systematic approach to the problem of defining and implementing resource management for cognitive radios. The generic interfaces of ULLA and GENI provide us with the mechanism to collect data and apply actions while hiding the complexity of underlying system(s) from the behaviours. In effect the implementation of behaviour components is simplified and they are to a large extent capable of being independent from the target deployment environment. The support for distribution means the problem of control channels negotiate is transparent to the behaviours that require it, and information is easily exchanged between the terminals in the system, while the policy engine verifies and enforces the operational constraints.

With respect to performing the optimisation task, CAPRI provides us with a systematic approach to problem of defining goals i.e. by providing the direct reward function for our the reinforcement learning example. While, the action broker/resolver abstractions enable us to modularise the learning task. Further, we able to explicitly separate the individual resource management tasks and their coordination. For example, our prototype application used a composite utility function to define the action selection criteria for the two behaviour, we may experiment with hierarchical decompositions e.g. subsumption based controllers [4], by using CAPRI to define a new activation policy for the action resolver.

VI. Conclusion and Future Work

We have proposed a component-based architecture to help realise extensible and portable solutions to recourse management in cognitive radio networks. With our architecture, control and optimisation mechanisms that improve the network performance are developed as reusable components. Generic interfaces provide transparent access to several networking and radio parameters, thus enhancing portability. Distributed control and co-ordination mechanism and policy engine assist in other primary functionalities of the cognitive radio network. The core CRM manages the administrative and control functionality. We illustrated the effectiveness of our architecture through an example.

In this paper, we have focus largely on the implementation architecture, in future work, we aim to develop a rich and expressive language structures for CAPRI and the policy language. We also aim to explore definition and practicality of more demanding behaviour coordination functions, for example, when numerous applications co-exist and compete for the resources. Finally we aim to explore how more real-time behaviours can be supported by our architecture.

ACKNOWLEDGEMENTS

This work was financially supported by European Union (ARAGORN project). We also acknowledge the partial support from DFG and RWTH Aachen through UMIC-research center facility. We thank other project partners for their input towards realising the architecture.

REFERENCES

[1] I.F. Akyildiz, W-Y Lee, M.C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Computer Networks*, 50(13):2127–2159, 2006.

[2] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

[3] M. Bandholz, A. Gefflaut, J. Riihijarvi, M. Wellens, and P. Mahonen. Unified Link-Layer API Enabling Wireless-Aware Applications. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–5, Sept. 2006.

[4] R. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, Mar 1986.

[5] M.M. Buddhikot. Understanding Dynamic Spectrum Access: Models, Taxonomy and Challenges. In *Proceedings of DySPAN*, apr 2007.

[6] P. Costa, G. Coulson, C. Mascolo, G.P. Picco, and S. Zachariadis. The RUNES middleware: a reconfigurable component-based approach to networked embedded systems. In *Personal, Indoor and Mobile Radio Communications, 2005. PIMRC 2005. IEEE 16th International Symposium on*, volume 2, pages 806–810, Sept. 2005.

[7] D. Elenius, G. Denker, M.-O. Stehr, R. Senanayake, C. Talcott, and D. Wilkins. CoRaL–Policy Language and Reasoning Techniques for Spectrum Policies. In *Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 261–265, June 2007.

[8] The GNU Radio project. http://www.gnu.org/software/gnuradio/.

[9] J. Mitola and G.Q. Maguire. Cognitive radio: making software radios more personal. *IEEE Personal Communications*, 6(4):13–18, Aug 1999.

[10] Video Conferencing Tool (VIC). http://mediatools.cs.ucl.ac.uk/nets/mmedia/wiki/VicWiki.

[11] K.E. Nolan, T.W. Rondeau, P.D. Sutton, C.W. Bostian, and L.E. Doyle. A framework for implementing cognitive functionality. In *Proceedings of the 51st SDR Forum General Meeting and Technical Conference*, 2006.

[12] M. Petrova and P. Mähönen. Cognitive Resource Manager: A cross-layer architecture for implementing Cognitive Radio Networks. *Cognitive Wireless Networks (ed. Fittzek F. and Katz M.), Springer*, pages 397–422, 2007.

[13] M. Petrova, P. Mähönen, J. Riihijärvi, M. Wellens, and B. INFOCOM. Cognitive wireless networks: your network just became a teenager. In *Poster Session of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[14] Wireless Open Research Platform (WARP). http://warp.rice.edu/.

[15] C.J. Rieser, T.W. Rondeau, C.W. Bostian, and T.M. Gallagher. Cognitive radio testbed: further details and testing of a distributed genetic algorithm based cognitive engine for programmable radios. In *IEEE Military Communications Conference (MILCOM)*, volume 3, pages 1437–1443, Oct 2004.

[16] Adaptive Reconfigurable Access and Generic Interfaces for Optimization in Radio Network (ARAGORN), 2008-2010. FP7 ICT project, EU commission.

[17] P. Sutton, L.E. Doyle, and K.E. Nolan. A Reconfigurable Platform for Cognitive Networks. In *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, pages 1–5, June 2006.

[18] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *In Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 1999.

[19] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[20] BBN Technologies. The XG Working Group, 2004. http://www.ir.bbn.com/projects/xmac/working-group/.