CogWnet: A Resource Management Architecture for Cognitive Wireless Networks

Ismail AlQerm*, Basem Shihada*, and Kang G. Shin†
* CEMSE Division, KAUST, Saudi Arabia, {ismail.qerm, basem.shihada}@kaust.edu.sa
† Dept. of Electrical Engineering and Computer Science, University of Michigan, USA, kgshin@eecs.umich.edu

***Abstract*—With the increasing adoption of wireless communication technologies, there is a need to improve management of existing radio resources. Cognitive radio is a promising technology to improve the utilization of wireless spectrum. Its operating principle is based on building an integrated hardware and software architecture that configures the radio to meet application requirements within the constraints of spectrum policy regulations. However, such an architecture must be able to cope with radio environment heterogeneity. In this paper, we propose a cognitive resource management architecture, called `CogWnet`, that allocates channels, reconfigures radio transmission parameters to meet QoS requirements, ensures reliability, and mitigates interference. The architecture consists of three main layers: *Communication Layer*, which includes generic interfaces to facilitate the communication between the cognitive architecture and TCP/IP stack layers; *Decision-Making Layer*, which classifies the stack layers input parameters and runs decision-making optimization algorithms to output optimal transmission parameters; and *Policy Layer* to enforce policy regulations on the selected part of the spectrum. The efficiency of `CogWnet` is demonstrated through a testbed implementation and evaluation.**

***Index Terms*—Cognitive radio, spectrum switch decision-making, transmission parameters, policy enforcement.**

I. Introduction

The demand for mobile services and applications is rapidly saturating wireless channel capacities. In particular, license-exempt bands such as Industrial, Scientific, and Medical (ISM) bands are experiencing increased channel demand and contention. Cognitive radio (CR) [1] can potentially address this problem through better management of wireless spectrum. CRs were first introduced by Mitola as an extension for software-defined radio (SDR) [2] but with the capability to adapt the radio transmission parameters to the underlying channel conditions. Dynamic Spectrum Access (DSA) [3] is one application of CR that grants access of a licensed spectrum to a secondary user. Mitola's vision was to build a cognitive architecture that is aware of the radio environment and can allow access to parts of the spectrum using customized transmission parameters. In fact, most of the current research in CRs focuses on DSA without exploring CR's potential for improving the performance of a wide range of networks and applications. There are several challenges in the development of such CR architectures. The first is to find a communication channel that is free (from primary users) and meets QoS requirements. The second is unreliability of the links established between source and destination in a dynamic environment where channel conditions are continuously changing. Third, interference impacts the quality of transmission when several transmissions are active within the same domain. Meeting these challenges increases the complexity of cognitive radio architecture design.

Several examples of cognitive architectures have been reported [4], [5]. They either focus on design optimization while ignoring issues that impact the overall system performance, or only consider DSA as the main objective. They use proprietary interfaces for communication between stack layers and the cognitive architecture. Moreover, no standard approach exists to configure the transmission parameters. There are a few high-level architectures that target resource management. However, often these lack a testbed evaluation that can determine their efficiency. For example, the design of a cognitive engine in [6] aims to learn and adapt to radio environment changes by extending legacy spectrum-sharing techniques. This engine lacks certain functions, such as the use of broadband waveform, reliability, and support for QoS requirements. The work in [7] uses a genetic algorithm for optimal configuration of physical- and MAC-layer parameters, such as modulation, transmit power, number of sub-carriers, etc. The system consists of a cognitive engine that adjusts the parameters according to environmental constraints. However, this engine only considers physical- and MAC-layers constraints, thus limiting reconfigurability. In addition, this architecture is very complicated as it uses a genetic algorithm without proposing ways to reduce its complexity. Another design of a reconfigurable node was proposed in [8] to observe and reconfigure all radio parameters. The design is simple and flexible but does not involve the application-layer requirements in the channel-selection process. In [9], the cognitive framework considers the interference caused by the coexistence of primary and secondary users. It is based on collecting channel information from the physical layer and using it to reconfigure the radio. The MAC layer includes the management and channel-sensing modules. However, this architecture lacks the involvement of the upper layers in the radio-management process, making the resultant decisions inconsistent with the QoS requirements.

All of the above-mentioned architectures do not specify interfaces enabling the flow of information between the cognitive engine and the stack layers. In addition, there is no support for distributed cognitive systems that improve the overall performance of the network. In this paper, we propose and build a cognitive architecture called `CogWnet` and use it to realize full cognitive functionality as per Mi-

978-1-4673-5775-3/13/$31.00 ©2013 IEEE

tola's vision. CogWnet addresses the challenges raised in cognitive resource management and coordinates the cognitive functions to resolve any conflicts. We use a cross-layer design approach [10] which infers and extracts network configuration parameters from all layers of the network stack for use in the decision-making process. In addition, CogWnet includes the following characteristics that enable superior network resource management:

1) *Adaptability*: achieved through optimization algorithms that evaluate and select suitable transmission parameters to meet QoS requirements under varying environmental conditions.
2) *Portability*: allows CogWnet to act as a resource management architecture in various wireless environments.
3) *Cooperative Decision-making*: the ability to negotiate the transmission parameters selected by CogWnet in one terminal with other terminals of the same network domain. This enhances the feasibility of the decision taken and avoids conflicts with other terminals.
4) *Generic*: CogWnet is not tied to any specific decision-making algorithm. It is a generic architecture that can accommodate various decision-making algorithms.
5) *Modularity*: CogWnet is able to resolve conflicts between the challenge it may encounter and the function it performs.
6) *Policy-awareness*: ability to apply spectrum access policies relative to stakeholders, users, and authorities.
7) *Comprehensive*: CogWnet includes input from multiple layers in the network stack. This allows for optimizing the radio parameters as per the device requirements.

CogWnet consists of three layers: the *communication layer*, which is a group of generic interfaces to fetch and send the sensed parameters from various layers in the network stack; the *decision-making layer*, which is used to run optimization and machine learning algorithms in order to select the optimal transmission parameters; and the *policy layer*, which consists of a policy engine that ensures the network policy enforcement. In this paper, we describe the various challenges and design considerations associated with each of these layers. We then implement the CogWnet architecture on a SDR testbed and present its performance results confirming the functionality of the proposed architecture.

This paper is organized as follows. We describe the *CogWnet* architecture in Section II, including the core functions that meet resource management challenges. Section III is dedicated to the real-time operation of *CogWnet*. Section IV presents our evaluation of CogWnet. Finally, we conclude the paper in Section V with a summary and directions for future work.

II. CogWnet ARCHITECTURE

CogWnet aims to improve spectrum utilization and radio resource management in a flexible and convenient manner. It executes the following cognitive functions.

Spectrum sensing: the ability to find available parts of the spectrum. We employed energy detection [11] to sense the spectrum and detect channel occupancy. Our implementation on USRP-N210 uses the spectrum sniffing capability of the hardware. We integrated it with C++ data structures used to implement energy detection for spectrum sensing.

Spectrum access: the function used to allocate channels for data transmission. CogWnet employs frequency agility as a spectrum allocation tool [12]. It is the ability to switch the operating frequency based on channels availability. Spectrum agility ensures that the allocated channels are set within an acceptable level of interference between the spectrum owners and the secondary users.

Transmission parameters adaptation: the third function applied to comply with the available channels conditions and QoS requirements. We use a decision tree [13] as the decision-making algorithm for parameters adaptation. The decisions aim to maximize utility functions, which are scalar functions that can assume values between 0 and 1. They are calculated from network performance attributes such as throughput, delay, jitter, and packet loss rate in order to assess the quality of the channels and the links. CogWnet employs utility functions that are adaptive based on radio-environment awareness. The general definition for the utility functions used in our implementation is:

$$U_{general} = \prod_i (u_i)^{w_i} \tag{1}$$

where u_i is the network performance attribute and w_i is the weight for that attribute. The weighting factor of a given attribute is assigned dynamically according to the current environmental conditions. Since we consider multiple factors such as throughput, interference, and packet loss, a generic utility function is proposed for each attribute according to the following equations:

$$\begin{cases} u_i = 1 + \tanh(\log(\frac{x}{x_0})), \text{if } \ x < x_0 \\ 1, otherwise \end{cases} \tag{2}$$

$$\begin{cases} u_i = 1 + \tanh(\log(\frac{x_0}{x})), \text{if } \ x > x_0 \\ 1, otherwise \end{cases} \tag{3}$$

where x is the average performance attribute value and x_0 is the threshold for the corresponding performance attribute. The utility in (2) is used to maximize the performance attributes such as throughput. (3) is used to minimize performance attributes such as packet loss and interference.

The last function executed by CogWnet is negotiation of channel conditions and the optimal transmission parameters selected during the adaptation with other nodes in the same domain. This cooperative function enhances the decision made for parameters configuration and avoids collision with the other nodes contending for the same channel. For example, CogWnet is able to determine through negotiation whether to assign the same, or two far apart channels to two contending pair of nodes. Consequently, communication can be established in a seamless manner while avoiding interference between the two nodes.

CogWnet is a container-based architecture. Each layer consists of a group of components. These components are composed of containers and exchange data through well-defined interfaces. A container is a data structure that contains a set of rules to be executed. These rules are defined as event, pre-condition, action, and post-condition. Rules statements are stored in the container as an ordered list representing their execution order. The event starts upon availability of data in the container based on a notification received from the communication layer. The pre-condition is a Boolean function to be evaluated for the execution to occur and it provides a condition to activate the containers. Post-condition is used to carry operation signals between containers (*i.e.*, callback function). All the containers in the framework communicate using connectors. The connector is a data structure that uses read/write commands to send data between containers. The containers notify the connector about data availability for communication. In addition, CogWnet includes another set of data structure called *modules*. The module is a structure with more than one container connected to process data flows in the architecture.

We now describe the CogWnet architecture components in details, including the communication layer, decision-making layer, and the policy layer.

A. Communication Layer

The communication layer abstracts information from other layers in the network stack and presents them to the decision-making layer. It consists of interfaces and channels that exchange control signals to collect application requirements and radio environment conditions. It uses three interfaces to collect data from the TCP/IP stack: universal link layer API (ULLA) [14] or any C++ based spectrum sensing data structure, generic network interface (GENI), and common application requirement interface (CAPRI). In addition, common control channel (CCC) [15] is used to exchange control messages among nodes for transmission parameters negotiation. ULLA is a generic interface to support access to MAC and physical layers parameters such as signal to noise ratio (SNR), received signal strength and frequency. In addition, It sends back selected transmission parameters to the corresponding stack layers to restrict the operating frequency range, adapt modulation scheme and adjust transmission power. All physical and MAC layer data extracted by ULLA is stored temporarily in the ULLA core before being fetched by CogWnet modules. The purpose of GENI interface is to support communication between CogWnet and transport and network layers. GENI accesses TCP/UDP data flows and determines loss ratio according to the number of received acknowledgments at the transport layer and routing information at the network layer. GENI and ULLA use the same data structure, which is built using UQL query language. CAPRI is a generic interface for applications to impose their QoS requirements and preferences to CogWnet decision-making layer. Utility functions are used to express the QoS requirements and CogWnet aims to maximize this utility using optimization algorithms. The following equation illustrates how application requirements can be expressed using utility functions:

$$U = 10 * log(t + 1000) + 3 * d \text{ where } (d \leq 0.050) \quad (4)$$

where U is the utility, t is throughput and d is the delay. This equation implies that the application cannot tolerate more than 50ms delay and that throughput has higher weight than delay with a weighting factor equal to 10. The utility function is registered in the core of CAPRI for later evaluation for CogWnet's decision-making. The common control channel is dedicated to exchanging transmission parameters between nodes. It can be utilized to negotiate the decisions made, improve spectrum usage, reduce network computation, discover neighbors and facilitate the detection of primary users in case of DSA.

B. Decision-Making Layer

The decision-making layer is the core of CogWnet as it accounts for receiving the sensory input from the communication layers and applying optimization algorithms (decision tree in this paper) to select the transmission parameters that improve the cognitive network performance. It consists of two components: repository and parameter mapper.

1) Repository: The repository is an entity that stores the sensory data received from the generic interfaces for further processing. It also schedules the various transmission parameter configuration decisions and sends it through the stack layers via the corresponding interfaces. The repository is composed of four containers and two modules as shown in Figure 1. Each container corresponds to different data collected from a distinct interface. The set of containers include:

- *Utility Manager*: stores and evaluates the utility function calculated by the CAPRI interface to extract the QoS requirements.
- *Link/Flow Manager*: stores the data collected from the network adapter about the traffic flows and the available links. This data is sent via GENI and ULLA interfaces.
- *Physical Container*: saves the data fetched from the spectrum sniffer through the C++ spectrum sensing data structure.
- *Negotiation Container*: stores queries and commands to be sent to remote nodes to share the transmission parameters selections based on the demand of the parameter mapper.

The following repository's modules reside in the user space to process the sensory data using cross-layer design, and schedule the decisions made.

- *Sensory Processing Module* is a filter-based on cross-layer design that combines all the data stored in the repository containers in the kernel space and sends them to the parameter mapper for decision-making. Moreover, it includes a database where the information about the existing scenario is saved for the purpose of learning.
- *Action Module* is a filter used to receive the selected transmission parameters from the parameter mapper, distribute

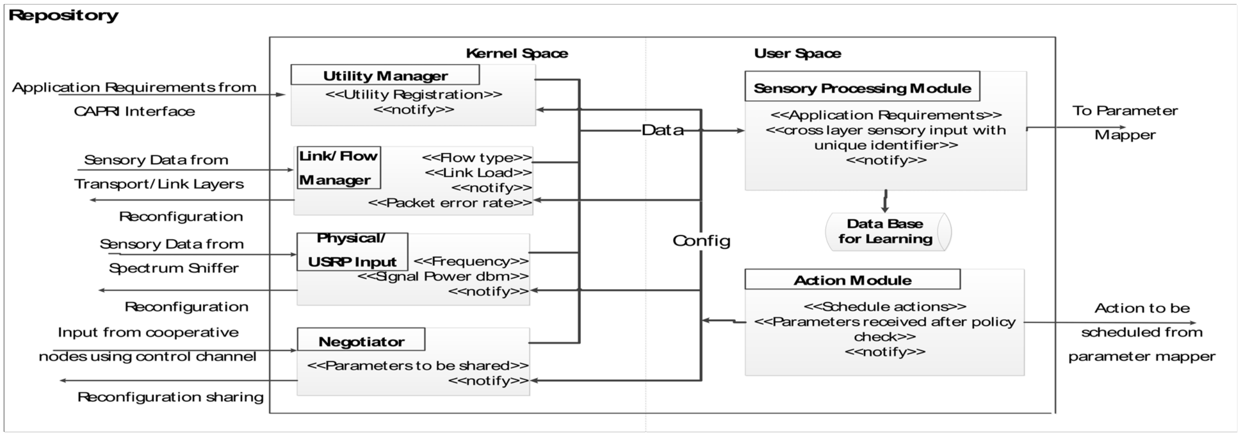

Fig. 1. Repository

them to the appropriate containers in the kernel level, and sends them back to the generic interfaces through those containers.

2) Parameter Mapper: The parameter mapper realizes the decision-making process in `CogWnet`. It consists of two modules and one container as depicted in Figure 2. The container is basically a temporary storage for the data flow between the parameter mapper modules. The modules that reside in the parameter mapper structure are as follows.

- *Action Broker Module* acts as a coordinator for the tasks executed by the decision-making layer. It receives the sensory input from the repository and sends it to the decision module. In addition, the action broker sends policy queries to the policy layer to check if the transmission parameters selected are compatible with the policy regulations. After receiving the policy-layer response, the action broker conveys the decision made to the action module in the repository.
- *Decision Module* receives requests from the action broker and take the following actions. First, check if there is any priority in case more than one channel requests are pending. Second, the channel allocation process starts to find the set of free channels according to the spectrum sensing information provided by spectrum sniffer. Finally, the decision module runs the decision-making algorithm and outputs the optimal transmission parameters back to the action broker for policy check.

C. Policy Layer

Stakeholders, users, and operators may be interested in applying certain constraints on the spectrum operation. For example, the operator can fix the range of usable frequency bands, so the decision made should not exceed that range. Therefore, the policy layer is required to enforce those constraints whether they are static or dynamic based on the geographical location. The policy rules are specified using declarative languages such as CoRal [16]. The policy layer in Figure 2 consists of the following three components.

- *Policy Server* stores all the spectrum policies and it is shared by all the nodes in the network to receive the set of rules that suit their queries. It consists of a server manager and a database. The server manager is the gateway of the server. It registers the nodes in the same domain and accesses the database to fetch policy rules and sends them to remote policy engines. The database stores the registration of all policy engines with their corresponding policy regulations.
- *Policy Engine* is the main component in the policy layer. It is responsible for checking whether the selected configuration parameters match the policy regulations. The policy engine is composed of a manager, a reasoner, and a database. The manager registers the policy engine in the server, receives all the policy regulations, and processes the policy queries. The reasoner checks the policy query and replies whether the selected configuration complies with the policy rules. It replies with a boolean value of 1 (‘Yes’) if the configuration is compatible with policy regulations, or 0 (‘No’) otherwise. The reasoner suggests a reason and a solution in case of a ’No’. All the policy rules that correspond to a certain spectrum class are stored in the database.
- *Trigger Manager* is a ring that connects the policy layer with the decision-making layer. It has two functions. First, it coordinates between the policy engine and the parameter mapper to check whether the selected configuration fits the corresponding spectrum policy. This function occurs in the policy module of the trigger manager. Second, it tracks the wireless channels status and informs the decision-making layer in case the channel is occupied. This function is executed in the detection module.

III. `CogWnet` IN ACTION

This section describes the procedure followed by `CogWnet` to allocate suitable communication channels and to adjust transmission parameters to improve the quality of wireless links. `CogWnet` constantly monitors the wireless environment and checks for channels availability using USRP-N210 sniffers [17]. Channels are considered free if the degree of interference experienced at the communication layer by the spectrum owners is lower than a specified threshold. If a channel is free, the communication layer abstracts the channel information and sends it to the decision-making layer. Each interface in the

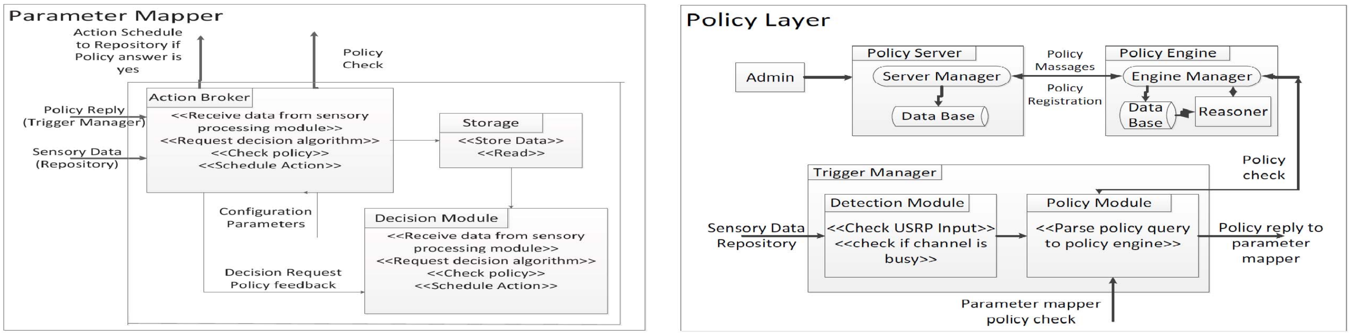

Fig. 2. Parameter Mapper and Policy Layer

communication layer accesses the corresponding layers in the stack to collect environmental parameters. At the same time, the sniffer keeps monitoring the interference levels. It prompts the decision-making layer to switch to a different channel if spectrum owners access the channel. The repository and the parameter mapper work closely to make environmental information available for the decision module. Then, the decision module allocates the most suitable channel using the concept of spectrum agility [12]. Next, the optimization process adapts the transmission parameters in order to match the environmental conditions. The output of the optimization process comprises the sets of parameters to be configured for all the stack layers. These parameters include flow rate for transport layer, transmission range, and number for hops for network layer, packet size and contention window size for MAC layer, modulation scheme and transmit power for physical layer. The selected parameters are then checked for policy compatibility using the policy layer. Subsequently, the negotiation process starts with `CogWnets` running on all the nodes of the network to avoid any conflict between terminals. Finally, the transmission parameters configuration is applied to the corresponding layers using the communication layer. In addition, `CogWnet` architecture accounts for improving system capacity and signal quality variation. Adaptive modulation is used to match the modulation scheme with channel conditions. For channel allocation, the decision-making layer adjusts the modulation as a function of the distance of the spectrum owners in order to match the quality of the signal received. The distance between the cognitive user and the primary user is used to control the transmission power to avoid interference. Figure 3 shows the state diagram for all functions executed by `CogWnet` for resource management and spectrum allocation. The diagram consists of two terminals (transmitter and receiver) each equipped with USRP-N210 [17] as a spectrum sniffer and GUN radio [2] as a radio platform able to extract the physical parameters collected by the sniffer. In addition, `CogWnet` components are implemented in these terminals for the decision-making process, reconfiguration and action negotiation. Most of the states occur within the repository as a main component for processing sensory information, and application requirements abstraction. The parameter mapper's only function is to run the decision-tree optimization algorithm

Function	Definition
Application requirements collection	To collect the application requirements using CAPRI Interface
Historical data collection	To consult the database at the repository for similar encountered scenarios
Spectrum data collection	To collect spectrum data using USRP-N210 sniffer
Device discovery	To discover the target device of the network flow and collect device capabilities
Optimal configuration determination	To determine the optimal network configuration based on the requirements and environmental information
Possible link configurations	To compute multiple possible link configurations on a device
Find a new radio or channel	To search if there is another radio channel configuration that could meet application requirements
Monitor network behavior	To monitor the network behavior of different applications and try to identify possible bottlenecks during network transmission. In this case, the "determine optimal configuration" function is triggered
Configure new radio or channel	This is included in the link configuration function and is used switch into different radio access technology if it is necessary
Configure new link parameters (in case of change to new channel)	To change the current link parameters to those determined by CogWnet to accommodate new application requirements or network conditions
Link Configuration	To push the parameters negotiated by a device to the underlying network card to create a link
Negotiation	To negotiate common channel and its parameters with other nodes

TABLE I
COGNTITVE FUNCTIONS FOR RADIO RESOURCE MANAGEMENT

and output the optimized transmission parameters. The trigger manager handles all policy issues. The cognition process functions are illustrated further in Table 1.

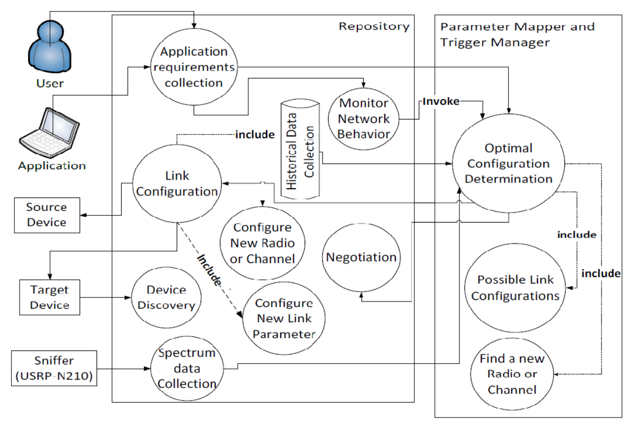

Fig. 3. Radio resource management: `CogWnet` modules execute cognitive functions. Actions are scheduled back through the link configuration state

IV. `CogWnet` EVALUATION

In this section we evaluate the functionality of `CogWnet` in terms of throughput and utility. `CogWnet` reduces packet loss by reducing interference and improves throughput. We use a home network with IEEE802.11a/g radio platform in this evaluation, given its heterogeneous nature in terms of type of devices and connection characteristics. Several channel impairments exist in this network such as interference and channel overloading. Therefore, there is potential for an architecture that manages home network and sustains its changes. The network topology used in this test consists of two pairs of terminals running `CogWnet`. Within each pair a different connection with different types of traffic is established. The wireless card of each terminal used in this evaluation is virtually divided into two cards, one for control exchange and negotiation and the other for data transfer. We use USRP-N210 hardware for spectrum sensing and the policy server is running on a separate computer.

The evaluation is running on Linux/windows based machines. We employed two kinds of applications in this scenario: UDP-based video streaming as a delay-sensitive application and normal TCP application. The UDP video stream application uses the first pair of nodes. The second pair has more than one application running. With legacy IEEE 802.11, all the transmitted data has to go through the access point which uses one common channel. All the nodes have to compete for this channel even if it is congested or interfering with other sources, resulting in throughput degradation and network overload. In addition, IEEE 802.11 is not capable of supporting different applications if bandwidth demand exceeds channel capacity. With `CogWnet`, however, the decision-making layer checks whether there is the possibility to establish a direct connection on a different channel between any of these pairs, based on the sensory information received from the communication layer. In this way, the throughput of the connection increases and does not compete with other connections that go through the same access point. `CogWnet` adjusts the channel width, center frequency, modulation order, symbol rate, transmission range for each hop and flow rate at transport layer to ensure that the capacity of the selected communication channel is sufficient to accommodate different QoS applications and switch to another channel if necessary. In this scenario, the sensory input from the communication layer includes ratio of successfully received packets, link load, received signal strength, packet error rate, spectrum occupancy power and application utility. Figures 4(a) and 4(b) show the goodput and the utility achieved by the home network with `CogWnet` enhancements. In the first period of 20 s, both pair of nodes run the same UDP video streaming with a data rate of 3 Mbps. Therefore, `CogWnet` decides to colocate both connections on channel 6 with a bandwidth of 6 MHz which is sufficient to accommodate both streams. After 20 s, the second pair decide to send two more TCP flows, one for high definition video and another for file download. Consequently, the communication layer detects high link load in the driver and notifies the repository. The decision module is informed by the repository and runs the utility-based optimization that moves the second pair of nodes to channel 11 and increases channel width to 20 MHz to satisfy the application requirements. Figure 4(b) shows that `CogWnet` manages to keep the application utility high regardless of the presence of different traffic flows from different nodes. Another test is carried out to evaluate the advantage of setting up a direct connection between nodes versus communication through an access point. Figure 4(c) shows the throughput achieved by the direct connection established by `CogWnet` compared with IEEE802.11 connection through access point. It is clear that the direct connection achieves better throughput as it is not affected by the distance from the access point or by contention with other nodes which send or receive their traffic through the access point. We also use interference as a performance metric to demonstrate `CogWnet` functionality. There are two possibilities for interference in the home network scenario: interference from another 802.11-CSMA based node or interference from external non-802.11 based sources such as any kind of packet injector. The ULLA interface in the communication layer monitors the received signal strength and link error rate. If interference affects the quality of the current transmissions, a signal is sent through the repository to the parameter mapper to reconfigure the radio to mitigate the detected interference detected. At the parameter mapper, the type of interference is determined according to the current channel characteristics. Figure 4(d) presents the performance of `CogWnet` in mitigating all kind of interferences mentioned in the established link in terms of throughput. The red line shows the case where there is a packet injector using channel 6. `CogWnet` observes this event and decides to switch the traffic to channel 11 to avoid interference and exploit full channel capacity. However, in the case where both type of interference exist (external source on channel 6 and 802.11 CSMA based source on channel 11, black line), the decision module colocates the traffic on channel 11 with a bandwidth of 20 MHz as no other non-overlapping channel is available. IEEE802.11 device experiences throughput degradation because it cannot switch to a non-interfering channel automatically.

Figures 4(e) and 4(f) present a comparison of throughput between 802.11 and `CogWnet` in the case of an external source interference and the case of both external and 802.11 based interference respectively. Initially, both frameworks obtain the same throughput as they use channel 6. Then, `CogWnet` starts the optimization process and switches all traffic to channel 11 as in Figure 4(e). In Figure 4(f), the decision-making layer colocates the traffic with the 802.11 based interferer on channel 11 and extends channel width to allow both connections to run with minimum interference. The 802.11 device stays in channel 6 with interference since it has no automated mechanism to change the channel. Finally, we evaluate `CogWnet` for reliability of transmissions. Received signal strength, propagation distance, and the number of successfully received packets are collected from the communication layer interfaces. Channel width, transmission range, and number of hops are tuned using the decision-tree optimization algorithm to reduce the packet loss ratio. Figure 5 compares the throughput obtained by 802.11 and `CogWnet` for various cases: sender and receiver are in the same place, in adjacent offices, in isolated offices or far away. `CogWnet` achieves better throughput and utilization as it adapts transmission parameters to minimize packet losses. `CogWnet` also outperforms 802.11 since it is difficult for 802.11 devices to predict channel and environmental changes.

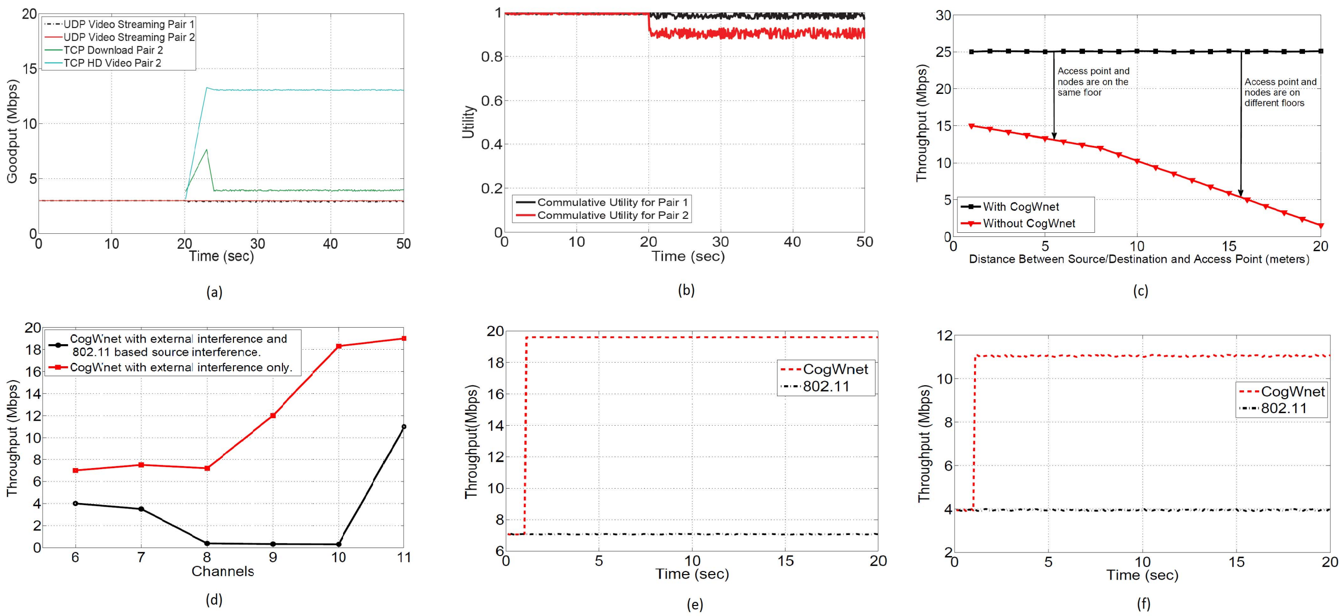

Fig. 4. Performance Evaluation of CogWnet

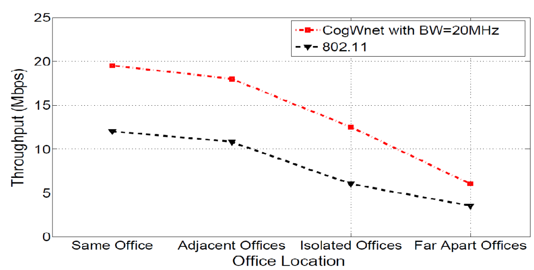

Fig. 5. Performance Evaluation of CogWnet (g)

V. Conclusion

We proposed `CogWNet`, a portable and flexible solution for radio resources management and spectrum optimization in cognitive radio networks. `CogWNet` is a generic architecture that senses the spectrum, allocates channels, and adapts transmission parameters using reusable components. For portability, generic interfaces are used to access the stack layers parameters. The cognitive functionality is enhanced by the distributed deployment. We have demonstrated the effectiveness of `CogWnet` with a real scenario in terms of the improved spectrum utilization and network throughput. We plan to conduct more implementation and experimentation for other types of spectrum such as IPTV and Long Term Revolution (LTE) networks.

References

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

[2] "The gnu project", http://www.gnu.org/software/gnuradio/.

[3] Z. Tabakovic, S. Grgic, and M. Grgic, "Dynamic spectrum access in cognitive radio", in *ELMAR*, September 2009, pp. 245–248.

[4] M. Petrova and P. Mhnen, "Cognitive resource manager: a cross-layer architecture for implementing cognitive radio networks", Cognitive Wireless Networks, Springer, 2007.

[5] Berk Canberk, Ian F. Akyildiz, and Sema Oktug, "A qos-aware framework for available spectrum characterization and decision in cognitive radio networks", in *PIMRC*, 2010, pp. 1533–1538.

[6] Feng Ge, Qinqin Chen, Ying Wang, C.W. Bostian, T.W. Rondeau, and Bin Le, "Cognitive radio: From spectrum sharing to adaptive learning and reconfiguration", in *Aerospace Conference, 2008 IEEE*, march 2008, pp. 1 –10.

[7] 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 *Military Communications Conference, 2004. MILCOM 2004. IEEE*, oct.-3 nov. 2004, vol. 3, pp. 1437 – 1443 Vol. 3.

[8] Paul Sutton, Linda E. Doyle, and Keith E. Nolan, "A reconfigurable platform for cognitive networks", in *Cognitive Radio Oriented Wireless Networks and Communications, 2006. 1st International Conference*, june 2006, pp. 1 –5.

[9] Yi Peng, Fenghong Xiang, Zengli Liu, Hua Long, and Jie Peng, "The novel cross-layer design for channel access in ad hoc cognitive radio network", in *Networking and Digital Society, 2009. ICNDS '09. International Conference on*, may 2009, pp. 217 –220.

[10] M. Bogatinovski and L. Gavrilovska, "Overview of cross-layer optimization methodologies for cognitive radio", 2008.

[11] Hano Wang, Gosan Noh, Dongkyu Kim, Sungtae Kim, and Daesik Hong, "Advanced sensing techniques of energy detection in cognitive radios", *Journal of Communications and Networks*, vol. 12, no. 1, pp. 19–27, 2010.

[12] Ashwini Kumar and Kang G. Shin, "Extended abstract: Towards context-aware wireless spectrum agility", in *in 13th ACM MOBICOM*, 2007, pp. 318–321.

[13] A. Navada, A.N. Ansari, S. Patil, and B.A. Sonkamble, "Overview of use of decision tree algorithms in machine learning", in *Control and System Graduate Research Colloquium (ICSGRC), 2011 IEEE*, june 2011, pp. 37 –42.

[14] Mahesh Sooriyabandara and etal, "Unified link layer api: A generic and open api to manage wireless media access", *Comput. Commun.*, vol. 31, no. 5, pp. 962–979, Mar. 2008.

[15] Brandon F. Lo, "A survey of common control channel design in cognitive radio networks", *Physical Communication*, vol. 4, no. 1, pp. 26–39, 2011.

[16] Qing Zhao and Brian M. Sadler, "Dynamic spectrum access: Signal processing, networking, and regulatory policy", *CoRR*, vol. abs/cs/0609149, 2006.

[17] "Usrpn210: Software defined radio", https://www.ettus.com/product/details/UN210-KIT/.