Cognitive Radio Experimentation World

Project Deliverable D7.6.4

Showcase of CNIT experiment

CABIN-CREW: The Wireless MAC Processor over CREW: Cognitive Access BenchmarkINg in CREW

Contractual date of delivery:	31-03-14
Actual date of delivery:	18-04-14
Beneficiaries:	CNIT
Lead beneficiary:	
Authors:	Ilenia Tinnirello (CNIT), Pierluigi Gallo (CNIT), Daniele Croce (CNIT), Fabrizio Giuliano (CNIT), Domenico Garlisi (CNIT), Michele Gucciardo (CNIT), Francesco Gringoli (CNIT), Nicolò Facchi (CNIT)
Reviewers:	Giuseppe Bianchi (CNIT), Hans Cappelle (IMEC)
Workpackage:	WP7 – Benchmarking the Federation
Estimated person months:	1.5
Nature:	R
Dissemination level:	PU
Version	2.1

Abstract: This deliverable describes the final CNIT showcase. We demonstrate the *effectiveness of the envisioned MAC programming interface* for defining and modifying MAC protocols that can be run on completely different hardware platforms (namely, the commercial Broadcom WiFi card and the advanced FPGA-based WARP board). The protocols can be interactively defined by means of a graphical interface and then compiled and injected in two competing nodes. A temporal trace of the channel idle and busy intervals is acquired for visualizing the programmed medium access operations. We also show *how to control these programming interfaces* by defining a MAC protocol adaptation strategy in OEDL language (to be executed by the Experiment Controller). The program is devised to improve the performance of WiFi and ZigBee overlapping networks, by introducing an inter-technology coordination mechanism in case of detection of interference problems.

Keywords: cognitive MAC schemes, MAC adaptation policies, hardware-independent MAC programming

REVISION HISTORY

Version	Date	Author	Description
1.0	10/04/2014	Fabrizio Giuliano	Initial draft with first inputs
2.0	17/04/2014	Ilenia Tinnirello	Integrations and revisions
2.1	18/04/2014	Hans Cappelle	Minor revision, final version

List of Figures

Executive Summary

This document describes the final demonstration planned for showing the integration of the Wireless MAC Processor architecture in the CREW testbed. The integration has involved two different planes: the *data-plane*, enabled by the availability of nodes supporting the WMP interface and able to change the data transmission mechanisms by specifying high-level hardware-independent programs; the *control-plane* built on top of the experiment control infrastructure, for coordinating the network-wide configuration of the node and collecting measurements.

Our showcase focuses on both the aspects. As far as concerns the demonstration of the programmable wireless data plane (section 1), we show how to interactively define and modify a MAC program to be injected in two competing nodes based on different hardware. The medium access behaviour of the two nodes is captured by the USRP nodes co-located in the remote testbed for debugging the MAC program directly over the air interface. For facilitating the definition of MAC protocols, we set-up a graphical programming tool in which the protocol state machine can be configured by using a classical drag and drop interface.

As far as concerns the complete demonstration of a MAC cognitive cycle (section 2), we propose an interesting use-case in which two independent technologies, namely WiFi and ZigBee, coexist and interfere in the same environment. We implement a control logic able to detect the coexistence problems and the MAC adaptation strategy. Since a programmable MAC architecture is available also for ZigBee nodes (the SnapMAC architecture [1], developed by other CREW partners), our control program works on both the ZigBee and WiFi nodes. Coexistence problems are revealed by monitoring the throughput performance and the error rates of the WiFi nodes. When the interference problem is detected, the central controller enforces a novel MAC program on the WiFi nodes and ZigBee nodes, based on a time-division access between the two technologies (i.e. WiFi and ZigBee nodes are entitled to transmit in different channel slots).

We conclude by remarking how the integration of **programmable software platforms** with the advanced **sensing and controlling functionalities** of the CREW testbed can significantly speed-up the prototyping of novel wireless protocol solutions.

List of Acronyms and Abbreviations

CABIN	Cognitive Access BenchmarkINg
CCA	Clear Channel Assessment
CREW	Cognitive Radio Experimentation World
DCF	Distributed Coordination Function
EC	Experiment Controller
GUI	Graphical User Interface
ISM	Industrial, Scientific and Medical
MAC	Medium Access Control
OEDL	OMF Experiment Description Language
OMF	cOntrol Management Framework
PHY	Physical Layer
PU	Primary User
SDN	Software Defined Networks
STA	Station
SU	Secondary User
RSSI	Receiver Signal Strength Indicator
TDM	Time Division Multiplex
USRP	Universal Software Radio Peripheral
WARP	Wireless open-Access Research Platform
WMP	Wireless MAC Processor
XFSM	eXtended Finite State Machine

Table of contents

1 Demonstrating MAC Programming and Portability

The goal of this demonstration is showing the effectiveness of the Wireless MAC Processor architecture [2] for implementing MAC protocols defined in terms of high-level programming languages. We integrated two different prototypes of the Wireless MAC Processor architecture on the CREW testbed [3]: the prototype working on commercial WiFi cards by Broadcom (already available before the starting of the project) called WMP-BROADCOM, and the prototype developed within CABIN-CREW project for the WARP software-defined-radio board called WMP-WARP.

Figure 1 shows the architecture of the demonstration platform, which includes the network programmable nodes and the user interface available for interacting with the system.

The **network** is composed by two programmable wireless nodes based on different hardware, namely WMP-BROACOM STA and WMP-WARP STA, attached to a WMP-BROADCOM AP.

The user interface is given by an input and output GUI, for controlling the demonstration and visualizing the results. The **control GUI** provides a set of tools for designing a MAC program (called MAClet) in terms of state machines, and for compiling the program in a machine-code (called bytecode) readable by the WMP prototypes. The **output GUI** is given by a web-interface connected to a USRP node for sampling the channel idle/busy state and visualizing it in a temporal trace. Channel traces can be useful for giving an immediate evidence of MAC real-time operation and re-configuration, without relying on indirect performance figures.

Figure 2 shows the layout of the control GUI organized into two main frames: the left frame, containing the global parameters of the state machine; the right frame where the graphical state machine is composed by linking logical states with specific transitions. Each state has a number of outgoing transitions triggered by the occurrence of events and enabled by the verification of an optional condition taken from a pre-defined list (i.e. from the WMP programming interface). Using conditional transitions allows to reduce the state space, since it decouples the actual state of the program into a logical protocol state (explicitly represented in the transition graph) and a configuration state (tracking the hardware state and the global program parameters). In our editor, we chose to implement Mealy state machines, in which the transition action is executed only if the transition is activated (i.e. upon the occurrence of the events and the verification of the condition).

Figure 3 shows an example of output GUI, where the different RSSI samples captures by the USRP are organized into a temporal channel trace. Packet and ACK transmissions are identified by RSSI values much higher than the background noise (about -95dBm). The two different values (-70dBm and -62dBm) correspond to the different distances between the transmitters and the monitoring USRP. Moreover, the duration of the sample burst of similar values depends on the frame duration (thus resulting much longer for the data frames).

For demonstrating the possibility to easily redefine different MAC schemes, we work on incremental modifications of a simple state machine implementing a traditional contention-based access protocol. Modifications can be decided run-time, but interesting examples can be: i) modifications of the contention parameters; ii) changing of random access to deterministic TDM access. The modified programs will be compiled and reloaded on the two heterogeneous hardware nodes during a run-time traffic session, while the channel analyser will be used for immediately visualize the channel access results.

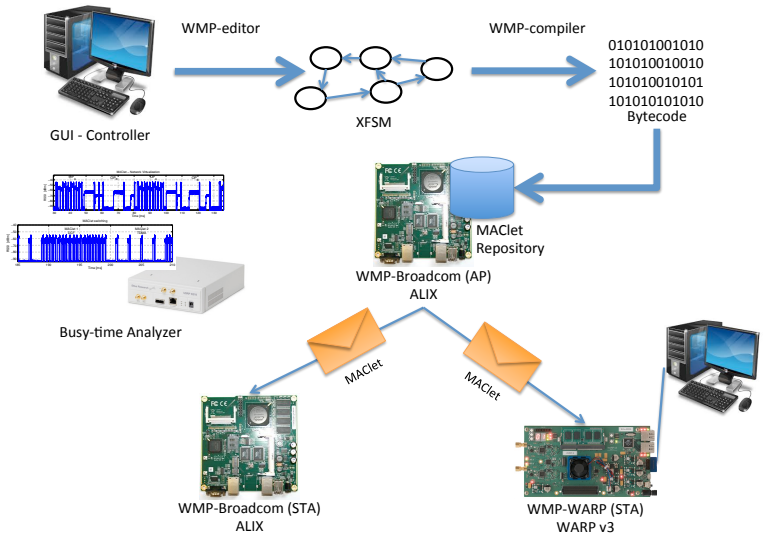

Figure 1 – Demonstration platform for MAC programming: user interfaces and network nodes.

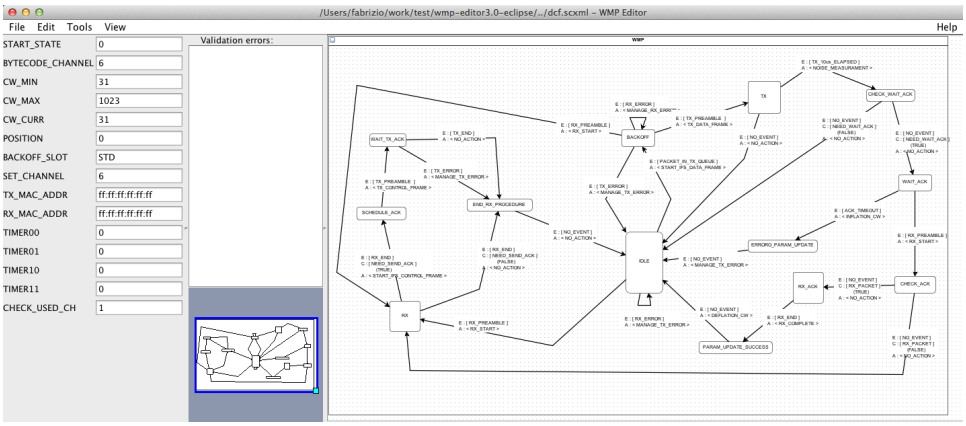

Figure 2 – Control GUI: Layout of the MAC Program Editor

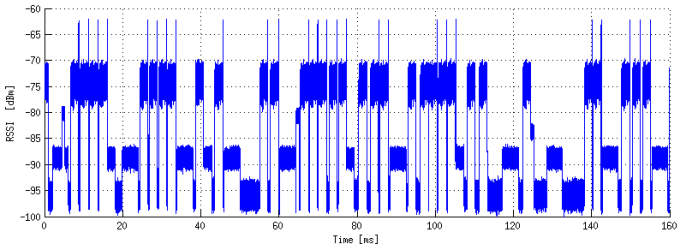

Figure 3 – Output GUI: A channel trace in which it is possible recognize packet transmissions, ACK transmissions and idle channel intervals.

2 Demonstrating a MAC Cognitive Cycle

The goal of this demonstration is exploiting the CABIN-CREW extensions for showing the support of a whole MAC cognitive adaptation cycle in CREW. We implement a detection and reaction strategy devised to avoid interference between WiFi and ZigBee technologies on unlicensed ISM bands. The experiment scenario is configured at the w-iLab.t testbed, with one pair of WMP-BROADCOM nodes and one pair of interfering ZigBee nodes, running TinyOS and the SnapMAC driver. For programming the network intelligence and enforcing the control commands, we use the cOntrol and Management Framework (OMF) and Measurement Library (OML). Although these frameworks have been designed for the configuration, execution and centralized control of experiments, they can be effectively used also for **network control** (as discussed in [4]).

We consider a MAC cognitive cycle in which: i) the sensing phase is implemented by collecting throughput and error statistics by means of dedicated monitoring applications deployed on the nodes; ii) the analysis and reasoning phases are performed at the Experiment Controller (EC) by aggregating data and defining customized events to be fired when inter-technology coexistence problems arise, iii) the adaptation phase is finally achieved by loading and/or activating inter-technology TDMA programs on the controlled nodes when needed. ZigBee and WiFi receivers report the achieved throughput to a central database using the OML framework.

Figure 4 shows the architecture of the demonstration. The **network** is composed of two WiFi nodes and two ZigBee nodes; two UDP data sessions are configured on both the links. A wired control network is used from piloting the nodes: the measurements are sent to the OMF database, while the control logic is implemented into a OEDL program executed by the EC.

The experiment **control interface** (web-based) is shown Figure 5. The interface allows to configure the ZigBee and WiFi traffic rate and to activate the two WiFi links. The same web interface is used for visualizing the **experiment output** by plotting the per-technology normalized throughput and the channel trace captured by the channel analyser.

For demonstrating the MAC cognitive adaptation cycle, we dynamically activate two links (under different traffic rates) and observe the throughput and channel access variations due the actions enforced by the experiment controller (EC). Figure 5 shows an example of these adaptations by comparing two channel traces capture before and after the detection of inter-technology interference. In absence of coordination, both the technologies experience throughput degradation. One of the major reasons of this performance degradation is the different granularity at which Clear Channel Assessment (CCA) samples are collected. The phenomenon is depicted in the top diagram of the figure, where it is evident that ZigBee frames (identified by an RSSI value of about -88 dBm) last for about 4 ms and that WiFi and ZigBee transmissions can collide. When the EC detects the problem, it loads an inter-technology MAC scheme on all the nodes. According to this MAC program, ZigBee nodes can transmit for a portion of the frame equal to 50 ms, while WiFi nodes can access the channel in the following 50 ms. The bottom diagram of the figure shows an example of coordinated channel access. The synchronization mechanism implemented in the program works as follows: ZigBee nodes autonomously switch between active and idle intervals, while being synchronized to the ZigBee coordinator. WiFi nodes switch to the activity interval after the detection of a burst of consecutive ZigBee packets and go to idle at the expiration of a timer.

.

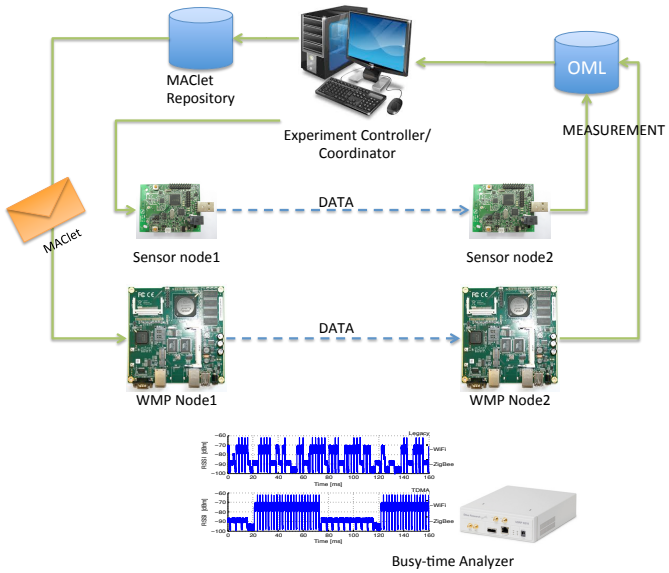

Figure 4 – Demonstration platform for MAC cognitive adaptations: user interfaces, control network and data network.

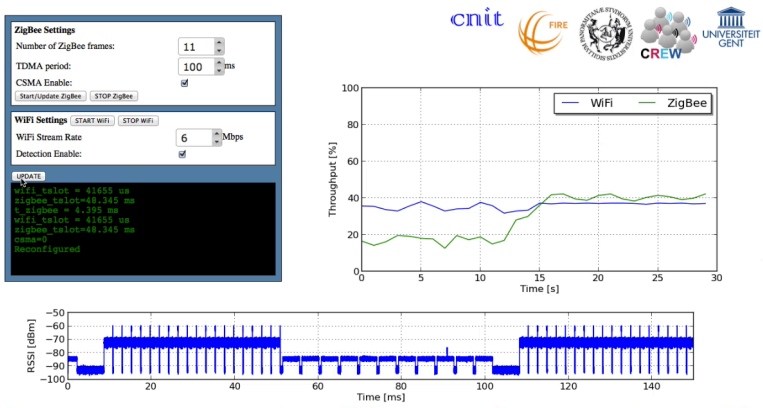

Figure 5 – Web-based user interface.

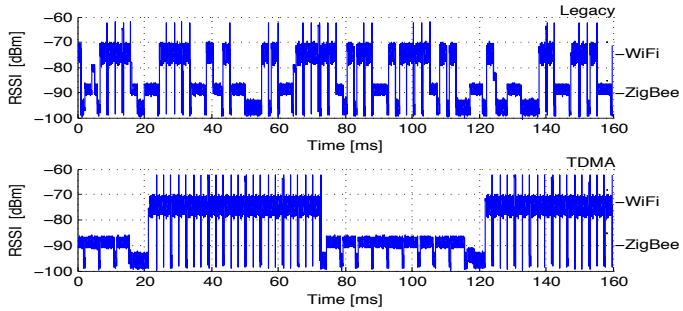

Figure 6 – Comparison of two channel traces before and after MAC adaptation.

3 Conclusion

The primary goal of our demonstration was the validation of the CABIN-CREW extensions for supporting MAC cognitive experiments. To this purpose, we developed and integrated several software tools on top of different hardware facilities already available in the testbed for: i) providing an **high-level interface for defining hardware-agnostic MAC programs** in terms of abstract state machines; ii) providing a **OMF-compatible control interface** for loading and activating MAC programs on the testbed programmable nodes.

Our experiments are targeted to demonstrate both the contribution and to enlighten how the joint exploitation of **programmable software platforms** and advanced **sensing and controlling functionalities** can significantly speed-up the prototyping of novel wireless solutions.

References

[1] P. De Mil, et al. “snapMac: a Generic MAC/PHY Architecture Enabling Flexible MAC Design.” Ad Hoc Networks (2014).

[2] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, “Wireless MAC Processors: Programming MAC Protocols on Commodity Hardware” IEEE INFOCOM, 25-30 March 2012

[3] Tinnirello, P. Gallo, M. Gucciardo, F. Gringoli, N. Facchi, D. Garlisi, F. Giuliano, Project Deliverable D7.6.3, “Final report on results of CNIT experiment and user experience”

[4] G. Bianchi, I. Tinnirello, “One Size Hardly Fits All: Towards Context-Specific Wireless MAC Protocol Deployment.”, in Proc. of ACM Wintech 2013.