SCIENCE AND TECHNOLOGY ORGANIZATION
CENTRE FOR MARITIME RESEARCH AND EXPERIMENTATION

**Reprint Series** **CMRE-PR-2019-024**

# Development of a Software-Defined and Cognitive Communications Architecture at CMRE

Roberto Petroccia, Giovanni Zappa, Thomas Furfaro, João Alves, Luigi D'Amaro

May 2019

Originally presented at:

OCEANS 2018 MTS/IEEE Charleston, 22-25 October 2018
doi: 10.1109/OCEANS.2018.8604849

## About CMRE

# Development of a Software-Defined and Cognitive Communications Architecture at CMRE

Roberto Petroccia, Giovanni Zappa, Thomas Furfaro, João Alves and Luigi D'Amaro

NATO STO Centre for Maritime Research and Experimentation
Viale S. Bartolomeo 400, 19126 La Spezia, Italy
email: {roberto.petroccia; giovanni.zappa; thomas.furfaro; joao.alves; luigi.damaro}@cmre.nato.int

***Abstract*—This paper presents an overview of the functionalities of the Cognitive Communications Architecture (CCA) that is currently under development at the NATO Science and Technology Organisation (STO) Centre for Maritime Research and Experimentation (CMRE). The CCA is designed to enable the deployment of advanced autonomous underwater solutions making use of smart, adaptive and secure underwater networking strategies. The CCA and the implemented network modules have been extensively tested, validated and improved during various at-sea campaigns conducted in recent years, involving CMRE and partners. The collected results show that the CCA is a robust, reliable and effective solution supporting underwater communications and networking. It provides various functions and services for the development of novel cognitive and secure communication strategies, such as a cross-layer networking functionality and the ability to easily integrate various existing communications technologies. These aspects are key enablers in the construction of a system that is robust to challenging conditions, such as those posed by varying channel conditions or adversarial attacks.**

***Index Terms*—Underwater communication, reconfigurable architectures, software defined networking, heterogeneous networks, network protocols, cross layer design, cognitive radio, cognitive communications architecture, CMRE.**

## I. Introduction

Increasingly novel and capable technologies are continuously being introduced into the field of underwater communications and maritime robotics. As the technologies mature, the possibility to explore novel operational scenarios becomes available [1]. One such scenario is the use of a network of cooperating unmanned and autonomous surface and underwater robots. The use of networked unmanned systems greatly reduces the human risk factor, and may reduce cost with an increased operational efficiency. The underlying connection between physically separated, unmanned systems is realized in a communications network.

The underlying physics upon which underwater communications relies are the limiting factor for performance. In the underwater environment, both radio and optical signals are greatly attenuated over short distances, with acoustic signals being the preferred modality for distances beyond about 50 m. Acoustical solutions suffer from long propagation delays (due to relatively slow celerity) and low data rates (due to high absorption at higher frequencies). Several other impairments can also affect acoustic transmissions and the quality of the received signals, including multipath, Doppler, frequency-dependent scattering effects [2].

Attempts at simply re-using communications solutions developed for the terrestrial domain have resulted low performance solutions, reflecting the challenge presented by the underlying physics. Novel solutions to create an Underwater Acoustic Network (UAN) have been proposed by the networking community, addressing single-hop and multi-hop scenarios composed by static and mobile nodes [3]–[5]. From the wide spectrum of solutions and approaches in literature, it is safe to state that there is no single approach fitting all possible scenarios. This comes essentially from the fact that the properties of underwater communication channels vary significantly in space and time.

Although most of the works have focused on the use of a single communication medium (i.e., acoustics) with limited cross-layering interaction, some development has been performed in the recent past on acoustic-optical underwater systems [6]. Aspects related to security have been marginally investigated (or not considered at all) leaving vulnerabilities open to potential exploitation. Furthermore, most approaches have only been validated and evaluated via means of simulations, with limited or no tests conducted at sea, thus reducing body of work proving reliability and robustness in real, harsh environments.

To overcome some of these challenges, CMRE has started the development of a hybrid, cognitive and secure underwater networking architecture, promoting the design, implementation and testing of novel software-defined communication strategies. The objective of this architecture, named the Cognitive Communications Architecture (CCA), is to enable the creation of an underwater system that is able to learn and make "smart" decisions regarding the communication technologies and configurations to use, thus adapting and reacting, in a distributed and ad-hoc way, to dynamic changes in the network.

Other underwater acoustic network frameworks and architectures have been proposed in the recent years [7]–[11], belying the increasing interest of the research community in novel solutions for UANs and their applications. Some of the proposed solutions are mainly simulation platforms on which novel protocols can be implemented and tested

This work was supported by the NATO Allied Command Transformation (ACT) Future Solutions Branch under the Autonomous Security Network Programme.

so that their performance can be evaluated. Others solutions exist to support the usage of real hardware to run in-lab experimentation and at-sea testing. Some of these make use of specific communication hardware while others support the use of multiple commercial off-the-shelf solutions. Most of the solutions currently available in the open literature, however, have been designed mainly to support research activities, ignoring important aspects and requirements of the operational community, including: 1) reliability and robustness requirements related to operational in-field activities (e.g., optimization with embedded hardware to reduce overhead, delay, or software dependencies of the whole system); and 2) security as a key consideration driving the system design (e.g., organization and protection of the information flowing across the stack or the interaction with third party software).

These important aspects have been considered from the design phase of the CCA, enabling the creation of a hybrid (heterogeneous and multi-modal), cognitive (highly adaptive and flexible) and secure (integrity and confidentiality) underwater network. Additionally, an efficient and modular implementation of the designed architecture has been developed, specifically addressing usage in field operations, deployment on various types of platforms with different computational and memory limitations, and interface with a wide range of hardware and software.

The rest of the paper is organized as follows. Section II introduces the Cognitive Communications Architecture proposed by CMRE, along with the main CCA functionalities and software components. Section III introduces the CCA protocol stack modules. The CCA implementation and dependencies are discussed in Section IV. The usage of the CCA during various at-sea trials is discussed in Section V. Finally, a final discussion with high-level conclusions are presented Section VI.

## II. The Cognitive Communications Architecture

The development and the design of the CCA have been derived from previous experiences with underwater communications and networking solutions at CMRE [12], [13]. These works present a conceptual design of the underwater communications systems, without an actual implementation. The CCA enhances and implements these designs, as a flexible solution which can be used by researchers to easily develop, test and validate specific algorithm and strategies for underwater communications. Additionally, the CCA supports quick “operational prototyping” for in-field experimentation. Following this approach, the overhead required to bring matured prototypes from the communications researchers to the more operational side is reduced, along with the resource requirement for maintenance and development.

Figure 1 shows the high-level CCA design and the major components, all of which follow the Software Defined Open Architecture approach described in [12]. The use of a layered structure maintains a separation between different layers, thus allowing the user to simply substitute the strategies to be adopted at a specific layer minimizing the possible impact on other layers. At the same time, the traditional Open Systems Interconnection (OSI) paradigm is enhanced by the possibility of using more than one approach (protocol module) at each layer of the stack, and the potential to use a cross-layer interaction mechanism. The selection of the specific solution(s) to use is driven by the presence of cognitive capabilities (policy engines), enabling the stack to autonomously reconfigure and adapt to the environmental or operational picture.

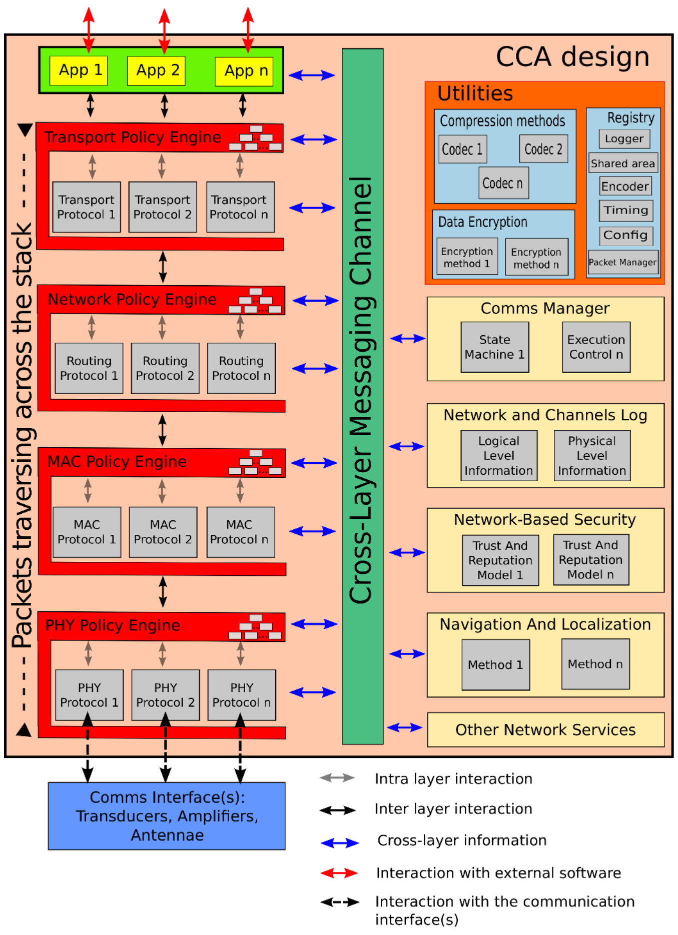


Fig. 1: The CCA design and its major components.

The protocol modules and policy engines represent the key component of the CCA. Data is transmitted in the form of packets traversing across the stack, while a cross-layer messaging channel is provided far a quick exchange of information among the various CCA modules and components. Details about the protocol modules and policy engines, along with a description of the additional modules used for the CCA operations, are detailed in the following subsections.

### A. *Protocol module*

The protocol module is the building block of the communications stack, with each protocol module providing some specific communications strategies for the specific layer it is part of. Protocol modules interact with and operate on packets through each layer of the stack. The specific protocol module used in each layer is selected by the policy engine of that layer.

Protocol modules do not interact with modules in the same layer, but generally have access to the contextual information provided through the cross-layer messaging channel. When a packet is traversing down the stack (during transmission), protocol modules may add headers to the packet which can contain protocol-module-specific “private” data that may be transmitted. Similarly, during reception, each module decodes its own specific header information embedded in the received message.

The CCA has been designed with the possibility of interfacing with “closed” modules. Although protocol modules should preferentially be open contributions to the community, the use of proprietary or sensitive solutions that require special protection should not be explicitly excluded. In this case the modules can be provided as black-box “compiled” solutions blocks. A common interface for these modules is envisioned similar to what is done for open implementations, likely inducing the need for writing a wrapping layer that converts between the native interface of the proprietary module and the open interface definition of the CCA.

This can provide a strong motivation for commercial partners to use the CCA as a federated open architecture in which their solutions can be deployed, and to contribute to the CCA development and related standardization efforts. In such an arrangement, providers of layer-specific solutions might explore new licensing paradigms in which they provide licensed, compiled binaries that implement CCA standard interfaces. Conceivably, modem vendors that currently sell hardware-based solutions could license their modulation schemes to be used instead on “organic” hardware of the particular CCA deployment and configuration. This concept can be extended to the different protocol layers, hopefully encouraging a general decoupling effort in the market promoting greater reuse and portability. Such an “app store“ concept could drastically change the vendor and signal lock-in that is widely experienced in the UAN device market today.

## B. Policy Engine

The selection of the protocols to use in each layer at any given instant is the responsibility of the corresponding policy engine of that layer. The policy engine can use contextual or performance information (or other mechanisms that embody the “cognitive” aspects of the system) to make this choice. The role of the policy engine is to select the appropriate protocol module(s) through which to route a packet received from the above layers for transmission[1]. A policy engine may decide to duplicate packets and to send multiple copies down the protocol stack, following different paths, e.g., if more than one communication medium has to be used for the same packet. Conversely, during reception, the policy engine must route the packet back through the paired protocol module that originated the packet on the transmitted node. Similarly to the protocol module, each policy engine can encode additional information in its own header/footer, if required, to pass data to the receiver nodes. The decision of how to route a packet through the stack and how to select the protocol modules to use is the key enabling feature for the implementation of cognitive capabilities.

[1]It is also possible for a packet to have a “route specification” (defined at the time of the generation of the packet or by the traversed layers) describing the intended route for that packet through the stack. This allows for a configurable soft-coupling to occur between layers that may have interdependencies.

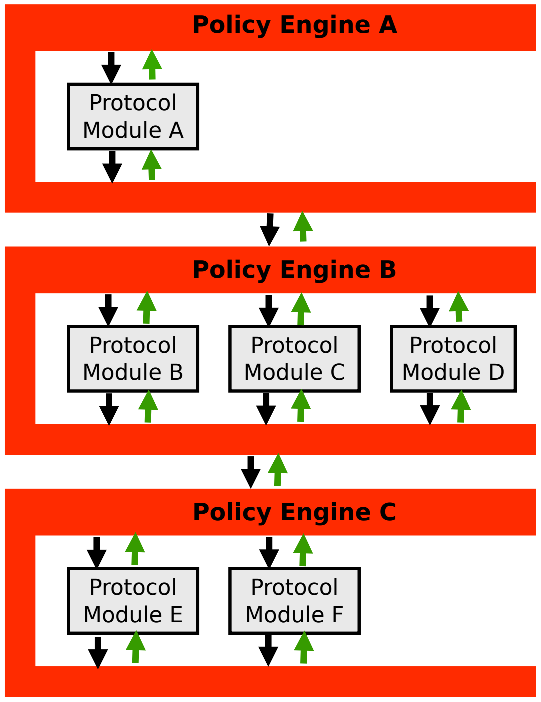


Fig. 2: Example of a three-layer protocol stack, black connections are for outgoing packets while green ones are for incoming packets.

Figure 2 shows a possible protocol stack configuration, with three layers, each with a different numbers of protocol modules at each layer. Each policy engine is connected to the one of the upper and lower layers, if available. Additionally, the protocol modules at each layer are connected to the policy engine of that layer. Using this design, each policy engine is able to first process each incoming/outgoing packet before passing it to the appropriate protocol modules. Moreover, after being processed by the protocol module, each policy engine can perform additional processing on the given packet in order to decide how it might need to be forwarded to the upper/lower policy engine.

The role of a policy engine as the supervisor for an entire layer raises the question about the concentration of layer-specific intelligence. At the two extremes are layers with policy engines that have complete knowledge of protocol operation and performance, with less capable protocol modules, versus highly capable protocol modules with a relatively unintelligent policy engine. In the former case more capable policy engines must be designed and implemented in order to handle all modules that may be configured. This increases the complexity of the policy engine. In order to enable additional flexibility and reduce this complexity, the use of hierarchical policy-engine "trees" is envisioned, as also depicted inside each policy engine layer in Fig. 1. A set of capable policy engines can be provided, each of them supporting a subset of the protocol modules available at that layer. Higher layers of the policy engine tree can then select the specific engine and associated modules to be used over time. A possible example of physical layer policy engine hierarchy is displayed in Fig. 3 where dedicated policy engines are designed for various communication mediums, which may have different needs. Each dedicated policy engine is tasked to select the protocol module(s) to use among the available ones, e.g., low/medium/high frequency radio or acoustic devices, optical platform and device configuration to use. A higher level policy engine is in charge of selecting the communication medium to use and then passing the control to the selected policy.

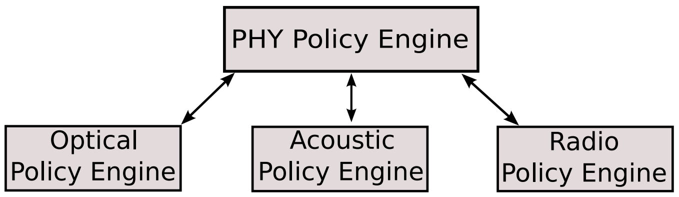


Fig. 3: Example of physical layer policy engine hierarchy.

### C. Data sharing

The sharing of data in the CCA represents one of the backbone capabilities of the architecture. The exchange of relevant information allows the different layers and components of the CCA to work in a coordinated and effective way. There are three ways data moves within the CCA: 1) data packets traversing across the stack; 2) cross-layer messaging channel; and 3) shared areas of memory.

*1) Packet:* A packet is the elemental data unit traversing through the stack in transmission/reception. A basic packet is composed of a common structure of key-value pairs where modules in the stack may read or insert data, and a payload (see Fig. 4). The key-value map in the common area is meant to serve as an exchange area for modules interacting with the same packet. This allows different modules operating on the same packet to reduce their individual overhead by using a "public" data component that is shared. At the same time, each of the traversed solution can add its own data in the form of a header and/or footer. These headers/footers are "semi-private"[2] structures that allow modules to associate specific data to specific packets, which may be used in later operations (typically encoding). Multiple headers and footers can be added. At the receiver side, the packet traverses up the stack following the opposite path. Header, footer and common data is decoded accordingly. This packet composition provides a

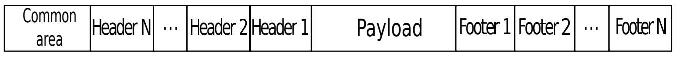


Fig. 4: The CCA packet.

natural separation between layers that lends itself to removal of inter-layer dependencies.

*2) Cross-layer messaging:* A cross-layer messaging interface provides a backplane over which modules can share contextual data regarding their operation and status. This mechanism is exposed via a publish-and-subscribe system, where modules can publish individual messages on a variable that may be read by any module that subscribes to updates for that variable. As a matter of practice, the cross-layer messaging channel is used to share information that is independent of individual packets.

*3) Shared area:* The shared area provides direct access to a set of variables (e.g., common settings) that can be modified by authorized modules. Read and write permissions are given to the various modules. This limits the access to some variables and allows the user to define the desired policies.

### D. Support modules

Additional functionalities that are used in support to the operations performed by the various modules include a *Clock*, *Timers*, *Logging* tools, a *Common dictionary* and *Data encoding/decoding* facilities.

*1) Clock:* This is the interface defining the way time is used in the CCA and how to transform it in different formats. It allows the developer to dilate the speed at which the CCA operates, using a floating point scaling factor. This functionality enables simulations with long real run times to be executed in a shorter period, and generally gives the user more flexibility when testing and debugging developments.

*2) Timer:* This `Timer` interface provides a timing utility to modules, including the possibility to start, stop, pause and check the status of a timer. The utility enables the execution of an arbitrary number of callbacks when the timer expires. The timer is also designed to be used in the case of periodic events (e.g., dumping the status of a module/variable every $n$ seconds). When used in this configuration, the timer implementation tracks and corrects any cumulative delay to provide a low-jitter periodic signal. All the timing operations are based on the `Clock` abstraction, preserving the time dilation feature.

*3) Logging:* The `Logger` provides an interface to log information when using the CCA. Information can be logged to large number of outputs, including to the console and to file. Each CCA module has a reference to its own logger. Different log priorities can be defined per logger to allow fine-grained output configuration. When a given priority is set for the logger, information having the same priority or higher than the specified level are logged. Like most utilities in the CCA, the base logger interface can be implemented differently and

[2]Private in the sense that there are no public interfaces to other modules to the underlying data. There are, however, no guarantees that malicious modules in the stack cannot abuse the underlying programming language to gain access. The CCA in general does not purport to provide defense against malicious internal code.

dynamically loaded at launch time to provide more advanced logging facilities.

*4) Common dictionary:* The `Common dictionary` defines a common data structure for the information to be shared within the CCA. This data structure is used by all the modules to share information through the various inter-module communications channels. The `Common dictionary` is composed by `dictionary elements`, all using the same data structure. This makes the passage of information among modules easier and enables developers to take advantage of specific utilities available for the `dictionary elements`, e.g., the use regular expressions to search for data or parameters, comparison between values, and arithmetic capabilities. New `dictionary elements` can be easily defined by the user, as might be required by novel approaches, and may be defined in two different ways: 1) the implementation of code; or 2) the preparation of a configuration file describing the new element. The latter option is more user friendly and is typically preferred. The configuration file is parsed at compilation time to automatically generate the required code.

*5) Encoder:* The `Encoder` transcodes standard `dictionary elements`, such as those contained in a packet, to binary formats suitable for transmission via the medium of choice. Similarly to the `dictionary element`, the user can specify how to encode the required information via a configuration file. The configuration describes which components of the `dictionary element` to encode, how many bits to use, what precision to use, and which encoding strategy to use. The configuration files are loaded by the `Encoder` at execution time, eliminating the need for specific compilation based on the possibly-changing configuration. Additionally, the encoder is designed to give the user the possibility to load different encoding configuration files for each module. In this way different rules can be applied to the same `dictionary element` for the various modules, depending on the specific requirement. One of the key functionalities provided by the encoder and the packet interface is the capability of performing what is called *just-in-time* encoding. Since some delay can be introduced from the creation of a packet to its actual transmission[3], the just-in-time encoding feature allows the data to be encoded at the time of transmission, thus making it possible to have the most up-to-date, accurate information in the outgoing packet.

## III. CCA DEVELOPMENT: IMPLEMENTED MODULES

During the past years the technical design and implementation of the CCA architecture has been continuously validated and improved through in-lab testing and at-sea experiments. Novel components and services have been defined, implemented, tested and validated over time, enhancing both the CCA infrastructural design and implementation, and the suite of supported protocol solutions. The CCA is still under active development, focusing on the provision of novel adaptive and cognitive capabilities.

[3]Delays can be introduced by the specific solutions at the different layers, e.g., backoff when reserving the transmission channel.

In what follows, an overview of the various protocol solutions currently implemented in the CCA (approximately ordered by macro-layers) is presented. All these solutions have been extensively tested and validated at-sea during various experimental campaigns (see Section V).

### A. Application

This layer describes the protocol modules that generate data or interact with external software providing data.

**Control module**: this module allows the user to locally change the CCA configuration in real time. It allows users to: 1) enable/disable the transmission of data for a given module or all the modules; 2) change the rate of transmission/sampling of classes of messages; 3) change the internal configuration of CCA modules (policy engines and protocol modules); 4) change the level of debugging of a given module or all the modules. This module will be extended to add control capabilities, such as the possibility to reconfigure remote nodes (e.g. submerged assets that have no other control channels).

**Position**: this module collects and processes streams of data containing position information. Currently, GPS and AIS data in the form of NMEA messages are supported as well as comma separated strings containing 2D or 3D data (i.e., latitude, longitude and depth).

**Underwater AIS**: this module collects GPS and AIS information (provided by the Position module) and filters the received data in space and time to be transmitted to submerged assets. Unlike regular AIS, the depth and the type (i.e., ship, moored buoy, manned or unmanned vehicle, etc.) of the assets are included in the transmission. This allows submerged nodes to broadcast their identification and localization data to any other assets in communication range, both above and below the surface. This module is largely used for water space management, providing situational spatial awareness to assets in the network.

**Underwater METOC**: a module collecting Meteorological and Oceanographic (METOC) data to be transmitted to submerged assets (e.g., temperature and wind speed). The received information can be then processed, reported and visualized by the operator (in case of a manned asset).

**Support for distressed submarine operations**: this module collects position and vital data from a distressed submerged asset to be broadcast in the area. The information selected for transmission is based on the NATO ATP/MTP-57 [14] (Submarine Rescue Manual) and interactions with the operational community [15]. The received information is then reported to be visualized by the operator.

**Chat**: this module implements a text chat functionality enabling the exchange of messages with another node (user) or a group of nodes (users). In practice, this has been useful when coordinating operations between manned assets (e.g. surface ship and submarine).

**DIVE**: this module implements the Distributed Id assignment and topology discoVEry (DIVE) protocol [16]. It is a distributed and ad-hoc protocol to self assign node IDs in the network and to discover the network topology and other relevant information. Although DIVE has been designed to discover the network and self assign node IDs in the network, it can be also used as leader election/consensus protocol to take other types of decisions in the network, e.g., to define a hierarchical organization of the network or distribute task allocations.

**MOOS**: this module interacts with the Mission Oriented Operating Suite (MOOS) software [17] deployed on board various vehicles at CMRE. The transmission requests generated by MOOS are collected and executed by the CCA. Similarly, the CCA provides to MOOS all the data collected by the protocol stack modules that can be used to support MOOS applications, such as Autonomous Underwater Vehicle (AUV) localization and navigation algorithms [18].

**Neptus**: this is a module that interfaces with the Neptus graphical interface [19] to visualize updates about the position and status of the deployed nodes, including distress messages.

**DUNE**: this module interacts with the DUNE software [20] deployed on board the vehicles developed by the LSTS group at the University of Porto [21]. Currently, it collects position information from the vehicle to be included in the underwater AIS picture. Additionally, it provides the vehicle with received AIS and distress information. AIS data can be used in support of vehicle navigation and water space management. The distress data can be used when employing autonomous vehicles in support of submarine escape and rescue operations, such as during the search and localization phase. The integration of additional functionalities in support of unmanned vehicle operations is currently on-going.

## *B. Transport*

A **fragmentation** module has been designed and implemented for this layer. This module takes care of fragmenting a large data message into smaller fragments, so that generated packets comply with the specific Maximum Transmission Unit (MTU) for a given link, similarly to the fragmentation algorithm in the Internet Protocol. The fragments are then reassembled by the receiving node to recreate the original larger message.

## *C. Network*

A **flooding** routing solution has been developed to forward data in the network. Each node immediately re-transmits a received packet (unless it is a duplicate) without the need of any additional control messages. To reduce the number of possible collisions, the implemented solution lets a node wait for a random time before forwarding the packet. Additionally, the user can define a probability of forwarding and a maximum number of hops to be traversed by a packet, in order to reduce duplication. Moreover, it is possible to define a static route for packets addressed to a specific destination node.

## *D. Data link*

This section describes the solutions currently implemented in what would be classified as the Data link layer in the OSI model, relating to data queuing and Medium Access Control (MAC).

**Priority queue**: this modules implements a priority queue for the received messages[4]. For each combination of message type (identifying a specific class of messages) and source address, a different queue is used. For each queue various parameters can be specified including: 1) the priority of the message; 2) the maximum capacity of queue; 3) if a new message has to be preferred to the ones already in the queue; 4) if some delay has to be applied between two consecutive polling of packets from the same queue for transmission; 5) the maximum number of seconds a message can stay in the queue (time to live) before being discarded; 6) if the message can be combined with other messages and transmitted in a single packet, or if a stand-alone transmission is required. The requested queue information can be added at the time of the creation of the packet or before it reaches the queue module. This enables adaptation in the processing of the priority queue for different types of messages over time. Variations for a given type may occur due to the assigned task or execution status of a task, environmental parameters, the status of the node and of the network.

**ALOHA**: this module implements the ALOHA-based MAC protocol described in [22]. This is a simple protocol for channel access that has the advantage of low overhead, as it does not perform extensive handshaking to avoid collisions. When a node has a data packet to transmit, it first checks whether the channel is idle or busy. If the channel is free, it starts the packet transmission, while is the channel is busy, the node delays the transmission according to a back off mechanism (linear, in the current implementation).

**TDMA**: this module implements a Time Division Multiple Access (TDMA) MAC protocol. It allows several nodes to share the same communication medium by dividing the time into different slots. Typically each slot is assigned to a specific node, with the duration of all slots being equal. The nodes transmit successively, each node using its own time slot. The combination of the allocated slots is called a frame and it repeats over time. A fixed guard time is usually placed between slots reduce the risk of interference. The TDMA solution implemented in CCA enhances the typical TDMA design allowing the user to configure the TDMA frame in a more flexible way. The user can define the number of slots to use, the duration of each slot and of its guard time, the assignment of a slot to a single node, a set of nodes or to all nodes. Using this design, it is possible, if needed, to reserve more time and more slots for some of the nodes which have more data to transmit with respect to the others. Similarly it is possible to enable concurrent transmissions (for nodes operating in

[4]The CCA priority queue module was inspired by the pAcommsHandler queue available for MOOS [17].

different areas), and the use of a hybrid approaches: ALOHA-based solutions could be used in slots used by multiple nodes, while regular TDMA for slots assigned to a single node.

### *E. Physical*

This layer describes the solutions implemented in the CCA to interact with the supported low-level communication interfaces and the CMRE channel simulator.

**JANUS**: this module implements a driver to transmit and receive JANUS messages [23]. It interacts with the JANUS-based software responsible for encoding/decoding messages. The implemented driver converts the data contained into CCA packets into the format used by the JANUS software and vice versa. It currently supports all the available JANUS plugins: 1) regular payload transmission; 2) underwater AIS; 3) underwater METOC; 4) support in distress submarine operations; and 5) first contact and language switching.

**Evologics**: this module implements a driver to configure, transmit and receive message using the S2C Evologics modem [24]. Firmware version 1.7 (and compatible subversions) is currently supported, including the use of extended notifications and synchronous instant messages (to request a transmission to occur at a specific time). Regular and Ultra-Short BaseLine (USBL) modem models are supported.

**Radio**: this module implements a driver to transmit and receive messages using a Radio antenna accessible via a TCP, UDP or serial stream connections. When a UDP connection is used, it also supports the transmission of multicast messages.

**Channel simulator driver**: this module implements a driver to interact with a CMRE channel simulator, which is used for development, testing and debugging of new solutions. The channel simulator can be configured with different channel parameters and interference models, in particular those relating to end-to-end comms performance. The use of such simulators is a prototyping and derisking tool, allowing for quick first evaluations of system performance in controlled environments, where different network topologies and configurations can be easily exercised before going to sea.

### *F. Utilities and services*

Several utilities and services exist as part of the CCA design (the right column of Fig. 1) that support the core operation of the components (policy engines and protocol modules) that form the basic stack. These tools don't participate directly in packet handling, but instead exist as services or capabilities that may be invoked by those client components.

**Cooperative ranging**: this module implements a cooperative approach for range estimation in support of vehicle navigation [18]. This approach is based on two-way Time of Flight (TOF) measurements and aims at reducing the overhead and delays in obtaining updates, while efficiently scaling to large networks. No fixed interrogation scheme is assumed and no synchronized clocks or dedicated instrumentation is required. When a node wants to collect a range estimation, it transmits a ranging request in broadcast. All the nodes receiving the request can decide to reply. Replying nodes coordinate to avoid collisions at the requesting node introducing some random delay before sending their message.

**Crypto**: this module implements the encryption functionality. Currently, the symmetric key cryptography implementation uses the Advanced Encryption Standard (AES) in combination with the Galois Counter Mode (GCM) approach [25]. Different hash tag sizes may be selected for message authentication and integrity verification.

### *G. Policy engines*

Initial policy engines have been designed and implemented with basic selection policies, including: the usage of routes depending on the message type (statically configured or adjusted by the traversed module of the stack); the forwarding of copies of the packet to multiple protocol modules in case of multiple non-interfering communication media; the encoding of header information for correct processing at the receiver in case of a dynamic and adaptive selection; and the forwarding of messages received by a communications interface for retransmission by the available interfaces, based on rules defined by the user. The design of more adaptive, intelligent policies is on-going. This line of research also includes:

- the definition of parameters (e.g., environmental data, node status, energy budget, message priority) and settings to be shared among the modules for “smart” and adaptive selection;
- the way to use, process and fuse the collected information;
- how to adapt and when to change selection strategy in optimal ways;
- how to generalize and abstract the concept of module performance such that coupling between policy engines and protocol modules is minimized.

Additional details about the current activities policy engine design activities at the physical layer (decision-tree aided adaptive modulation) and for the integration/interaction of upper and lower layers of the stack can be found in [26] and in [27], respectively.

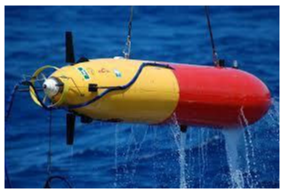
## IV. CCA IMPLEMENTATION AND DEPENDENCIES

The CCA is written in `C++14`. The core package design makes heavy use of the *bridge pattern* [28], promoting flexibility and reusability of components. From the early days of design, special consideration has been given to the separation of interface and implementation to both promote sharing and reuse but allow for intellectual property protection.

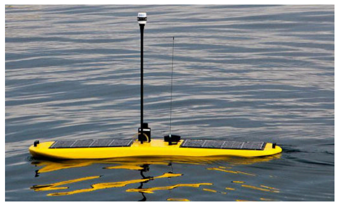
The CCA can be compiled and linked under a Linux environment using `CMake` [29] and a compiler compatible with the `C++14` standard. The CCA runs as single multi-threading program, where independent threads are used for each packet, cross-layer messaging interface, and timer interface. The CCA hides the complexity of the multi-threading implementation,

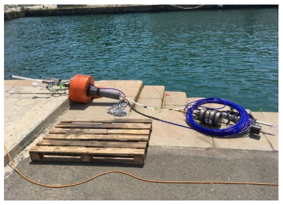
(a) eFolaga AUV.

(b) SPARUS AUV.


(c) OEX AUV.

(d) Gulliver ASV.


(e) Waveglider.


(f) Portable drifting gateway buoy.

(g) Moored buoy.

(h) Manta portable device.

Fig. 5: Some of the CMRE assets using the CCA in support to underwater communications.

allowing developers to focus on the creation of novel communications functionalities and solutions with the support of CCA backbone components. Backbone components are compiled as static or dynamic libraries while protocol strategies (i.e., policy engines and protocol modules) are compiled as dynamically linked libraries that can be loaded at run time as plugins.

In order to run the CCA only few additional libraries are required by the CMRE backbone implementation: `log4cpp` for logging and `libconfig++` for configuration file handling[5]. The configuration file used for both the `dictionary element` and the `Encoder` is based on the JavaScript Object Notation (JSON) [30] open-standard format, which is largely used by the community of developers.

The limited inherent dependencies of the CCA simplifies greatly the effort of cross-compilation for embedded systems. CCA has been successfully cross-compiled to run on various ARM-based embedded solutions, including Raspberry Pi2 [31], Raspberry Pi3 [32] and Beaglebone Black [33].

Additional dependencies may be introduced by specific modules. In order to use the cryptographic module, the encryption library `LibCryptopp` is required. The Automatic Identification System (AIS) and National Marine Electronics Association (NMEA) modules make use of libraries to parse the NMEA data. The use of the JANUS module obviously induces the requirement for the JANUS library [34] and its sub-dependencies (`Alsa` and `FFTW`).

To facilitate handling these dependencies, the CCA build ecosystem has scripts to compile the code that automatically searches in the filesystem for the required libraries. If a library is not found, the target module is not compiled and a warning message is reported to the user.

## V. CCA Experimental Results

Over the past four years, the CCA and the various implemented capabilities have been validated and evaluated extensively using the CMRE Littoral Ocean Observatory Network (LOON) and during various at-sea campaigns together with partners and collaborators, e.g., REP16-Atlantic, REP17-Atlantic, Dynamic Monarch 2017, and CommsNet17.

During these experiments, the CCA was successfully installed on different, heterogeneous CMRE assets (Fig. 5), including AUVs, Autonomous Surface Vehicles (ASVs), portable and moored units. Additionally, the CCA and its solutions were installed on board of partners platforms: the NRP Arpão submarine during the REP series; the ESPS Tramontana submarine and various rescue ships during the Dynamic Monarch 2017 exercise; two AUVs from FEUP (Fig. 6) during REP17-Atlantic.

Fig. 6: The LAUV vehicles from FEUP.

During the various experimental activities, the CCA was used to run all the networking experiments for the investigated solutions, mostly focusing on: 1) the DIVE protocol; 2) the cooperative ranging and networked long baseline (NetLBL) solution; 3) the use of JANUS-based services for Underwater AIS, METOC, chat and support of distressed submarine activities; 4) the testing of the crypto module as a service to other modules; and 5) the deployment of a persistent underwater

[5]According to the *bridge pattern*, the user could define and implement its own logging and configuration strategies without depending on these libraries.

network[6]. In all the considered scenarios and configurations the CCA architecture has proven to be flexible enough to enable the implementation and integration of the developed modules.

In what follows quantitative measurements to validate the robustness, flexibility and reliability of the implemented communication architecture are presented.

## *A. Metrics*

To validate and evaluate the performance of the CCA implementation, the following metrics have been considered:

- **CPU usage**: Average and maximum CPU usage during the conducted experiments. It is expressed as a percentage with respect to the total CPU capacity;
- **RAM usage**: Average and maximum RAM memory usage while performing the different experiments. It is expressed as a percentage with respect to the total RAM capacity;
- **Delay**: Average and maximum delay at the transmission and reception time. This delay includes the time for the packet to be encode/decode and processed by the different layers across the protocol stack. It is expressed in milliseconds.

These metrics have been selected to evaluate the additional load and overhead introduced by the proposed system to the networking operations. Being able to design and implement a lightweight architecture is important as such systems frequently must be deployed on small, energy efficient, compact and inexpensive devices. These devices can then be easily installed in small housings (e.g., modem casings and on autonomous underwater and surface vehicles), thus easing test preparation, and reducing costs and time related to deployment activities. Additionally, we consider these metrics to be early descriptors of the performance of the implementation itself as a software infrastructure — clearly metrics regarding the efficacy of a given communications infrastructure are part of an largely-orthogonal conversation relating to the specific protocols (or combination of protocols) deployed.

## *B. Results*

Various experiments have been conducted during the various sea trials having the CCA running on capable hardware as well as cheap, less-capable embedded platforms. In what follows we present the results for some of the organic platforms used for CCA operations:

**Laptop**: 4 CPU i7-2640M @2.8GHz and 8GB of RAM;

**SECO [38]**: x86 platform equipped with a Dual Core processor @1.0GHz and 4GB of RAM;

**RaspberryPi2**: ARM platform equipped with a Quad-core processor @900MHz and 1GB of RAM.

Table I shows the collected results for the different considered assets. All the values are averaged over many days of continuous operations with the transmissions and reception of tens of thousands messages, using both JANUS and Evologics modems.

| Platform | CPU Usage | RAM Usage | Delay Tx | Delay Rx |
|---|---|---|---|---|
| | Avg (max)% | | Avg (max) ms | |
| Laptop | 0.5 (1.3) | 0.4 (0.5) | 0.6 (2) | 0.9 (3) |
| SECO | 0.9 (1.8) | 0.6 (1.2) | 1.2 (2.5) | 1.7 (3.5) |
| RaspberryPi2 | 1.3 (2.0) | 1.1 (2.2) | 1.9 (3.2) | 2.1 (3.8) |

TABLE I: Quantitative measurements of the CCA implemented communications architecture running on various assets.

The collected results clearly show that the usage of the resources introduced by the CCA backbone modules and by the protocol solutions is minimal for all the considered platforms. This gives the developer headroom to run additional software (vehicle mission planning, data processing, sensor monitoring, etc.) on the same systems executing the CCA, without the requirement for additional or dedicated platforms. Similarly, the introduced delays are very low, especially when compared with typical delays of acoustic transmissions and propagation in water.

## VI. Conclusions and Future Works

This paper presented the CMRE Cognitive Communications Architecture. The CCA currently represents a robust, reliable and effective solution to support underwater communications and networking research. It provides various functions and services to support the development of novel, cognitive, and secure strategies, such as the cross-layer networking functionalities and the support for different available communications technologies. These aspects are key enablers to automatically adapt to varying channel conditions which may arise due to environmental factors or malicious activities. The CCA has been tested and validated during various sea campaigns on board of a variety of heterogeneous maritime platforms.

Although additional refinements can be investigated for the CCA backbone implementation, the main effort, and area of active CMRE research, will focus on the cognitive components of the CCA. The effort of CMRE to implement, test, validate and ultimately integrate the development of software and hardware components represent a step ahead in the direction of the creation of an advanced autonomous maritime communication system. This system has the potential to enable diverse and highly effective maritime operations.

One of the CMRE objectives is to foster standardization and interoperability. Therefore there is the intent to share the CCA open interfaces with the community in order to build a shared ecosystem towards the design and development of interoperable software-defined and cognitive solutions. It should be also possible for industrial partners to join this effort by providing licensed or otherwise closed solutions to be integrated as modules of the architecture, thus opening new market and collaboration opportunities.

[6]For the interested readers, details about the REP16-Atlantic activities can be found in [35], [36], REP17-Atlantic and Dynamic Monarch in [15], and CommsNet17 in [37].

## References

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: Applications, advances, and challenges," *Royal Society*, vol. 370, no. 1958, pp. 158–175, May 2012.

[2] M. Stojanovic and J. Preisig, "Underwater acoustic communication channels: Propagation models and statistical characterization," *IEEE Communications Magazine*, vol. 47, no. 1, pp. 84–89, January 2009.

[3] T. Melodia, H. Khulandjian, L.-C. Kuo, and E. Demirors, "Advances in underwater acoustic networking," in *Mobile Ad Hoc Networking: Cutting Edge Directions*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds. Hoboken, NJ: John Wiley & Sons, Inc., March 5 2013, ch. 23, pp. 804–852.

[4] K. Chen, M. Ma, E. Cheng, F. Yuan, and W. Su, "A survey on mac protocols for underwater wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1433–1447, March 2014.

[5] N. Li, J.-F. Martínez, J. M. Meneses Chaus, and M. Eckert, "A survey on underwater acoustic sensor network routing protocols," *Sensors*, vol. 16, no. 3, p. 414, March 2016.

[6] F. Campagnaro, R. Francescon, P. Casari, R. Diamant, and M. Zorzi, "Multimodal Underwater Networks: Recent Advances and a Look Ahead," in *Proceedings of the 12th ACM International Conference on Underwater Networks and Systems - WUWNet'17*. Halifax, NS, Canada: ACM Press, November, 6–8 2017.

[7] D. Torres, J. Friedman, T. Schmid, and M. B. Srivastava, "Software-defined underwater acoustic networking platform," in *Proceedings of the Fourth ACM International Workshop on UnderWater Networks (WUWNet '09)*, Berkeley, California, USA, November 3 2009, pp. 7:1–7:8.

[8] S. N. Le, Z. Peng, J.-H. Cui, H. Zhou, and J. Liao, "Sealinx: A multi-instance protocol stack architecture for underwater networking," in *Proceedings of the Eighth ACM International Conference on Underwater Networks and Systems*, ser. WUWNet '13, Kaohsiung, Taiwan, November 11–13 2013, pp. 1–5.

[9] M. Chitre, I. Topor, and T.-B. Koay, "The UNET-2 modem - An extensible tool for underwater networking research," in *Proceedings of MTS/IEEE OCEANS 2012*, Yeosu, Korea, May, 21–24 2012.

[10] C. Petrioli, R. Petroccia, J. R. Potter, and D. Spaccini, "The SUNSET framework for simulation, emulation and at-sea testing of underwater wireless sensor networks," *Ad Hoc Networks*, vol. 34, no. C, pp. 224–238, 2015.

[11] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, and M. Zorzi, "DESERT underwater: an NS-miracle-based framework to DEsign, simulate, emulate and realize test-beds for underwater network protocols," in *Proceedings of MTS/IEEE OCEANS 2012*, Yeosu, Korea, May, 21–24 2012.

[12] J. R. Potter, J. Alves, T. Furfaro, A. Vermeij, N. Jourden, G. Zappa, A. Berni, and D. Merani, "Software Defined Open Architecture Modem Development at CMRE," in *Proceedings of the 2nd IEEE OES International Conference on Underwater Communications and Networking*, ser. UComms14, Sestri Levante, Italy, September, 3–5 2014.

[13] J. Alves, K. LePage, P. Guerrini, J. Potter, G. Zappa, T. Furfaro, A. Munafó, and A. Vermeij, "Underwater communications research and development at CMRE," in *Proceedings of MTS/IEEE OCEANS 2015*, Genova, Italy, May 18–21 2015, pp. 1–7.

[14] NATO, *STANAG 1390 Ed. 8: The Submarine Search and Rescue Manual*. NATO Standardization Office, 2013.

[15] J. Alves, R. Petroccia, A. Grati, N. Jourden, G. Vitagliano, P. S. Garcia, J. N. Prieto, and J. Sousa, "A paradigm shift for interoperable submarine rescue operations: The usage of JANUS during the Dynamich Monarch 2017 exercise," in *Proceedings of MTS/IEEE OCEANS 2018*, Kobe, Japan, May 28–31 2018.

[16] R. Petroccia, "A distributed ID assignment and topology discovery protocol for underwater acoustic networks," in *Proceedings of the 3rd IEEE International Conference on Underwater Communications and Networking*, ser. UComms16, Lerici, Italy, August 30 – September 1 2016.

[17] "MOOS: Mission oriented operating suite," Last time accessed: March 2015. [Online]. Available: http://www.moos-ivp.org

[18] J. Śliwka, R. Petroccia, A. Munafò, and V. Djapic, "Experimental evaluation of Net-LBL: An acoustic network-based navigation system," in *Proceedings of MTS/IEEE OCEANS 2017*, Aberdeen, Scotland, June 19–22 2017.

[19] P. Dias, S. Fraga, R. Gomes, G. Gonçalves, F. L. Pereira, J. Pinto, and J. Sousa, "Neptus - a framework to support multiple vehicle operation," in *Europe Oceans 2005*, IEEE. IEEE, 2005.

[20] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. R. B. Marques, and J. B. Sousa, "The LSTS toolchain for networked vehicle systems," in *Proceedings of MTS/IEEE OCEANS 2013*, Bergen, Norway, June, 10–13 2013, pp. 1–9.

[21] "Laboratrio de Sistemas e Tecnologia Subaqutica (Underwater Systems and Technology Laboratory), University of Porto, Portugal," https://www.lsts.pt/, Last time accessed: March 2018.

[22] S. Basagni, C. Petrioli, R. Petroccia, and M. Stojanovic, "Choosing the packet size in multi-hop underwater networks," in *Proceedings of IEEE OCEANS 2010*, Sydney, Australia, May, 24–27 2010, pp. 1–9.

[23] J. R. Potter, J. Alves, D. Green, G. Zappa, I. Nissen, and K. McCoy, "The JANUS Underwater Communications Standard," in *Proceedings of the 2nd IEEE International Conference on Underwater Communications and Networking*, ser. UComms14, Sestri Levante, Italy, September, 3–5 2014.

[24] Evologics, "Evologics S2C acoustic modems," Last time accessed: March 2018. [Online]. Available: http://www.evologics.de/

[25] M. J. Dworkin, "NIST SP 800-38D. Recommendation for Block Chiper Modes of Operation: Galois/Counter Mode (GCM) and GMAC," Technical report, Gaithersburg, MD, USA, 2007.

[26] K. Pelekanakis, L. Cazzanti, G. Zappa, and J. Alves, "Decision tree-based adaptive modulation for underwater acoustic communications," in *Proceedings of the 3rd IEEE International Conference on Underwater Communications and Networking*, ser. UComms16, Lerici, Italy, August 30 – September 1 2016.

[27] R. Petroccia, K. Pelekanakis, J. Alves, S. Fioravanti, S. Blouin, and S. Pecknold, "An Adaptive Cross-layer Routing Protocol for Underwater Acoustic Networks," in *Proceedings of the 4th IEEE OES International Conference on Underwater Communications and Networking*, ser. UComms18, Lerici, Italy, August 28–30 2018.

[28] Wikipedia, "Bridge pattern," http://en.wikipedia.org/wiki/Bridge_pattern, Last time accessed: July 2018.

[29] "cmake solution," https://cmake.org/, Last time accessed: March 2018.

[30] "JavaScript Object Notation," https://en.wikipedia.org/wiki/JSON, Last time accessed: March 2018.

[31] Raspberry, "Raspberry pi 2," https://www.raspberrypi.org/products/raspberry-pi-2-model-b/, Last time accessed: March 2018.

[32] ——, "Raspberry pi 3," https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, Last time accessed: March 2018.

[33] Beaglebone, "Beaglebone black," https://beagleboard.org/black, Last time accessed: March 2018.

[34] "JANUS wiki," Last time accessed: March 2015. [Online]. Available: http://www.januswiki.org/tiki-index.php

[35] R. Petroccia, J. Alves, and G. Zappa, "Fostering the use of JANUS in operationally-relevant underwater applications," in *Proceedings of the 3rd IEEE International Conference on Underwater Communications and Networking*, ser. UComms16, Lerici, Italy, August 30 – September 1 2016.

[36] ——, "JANUS-based services for operationally relevant underwater applications," *IEEE Journal of Oceanic Engineering*, vol. 42, no. 4, pp. 994–1006, July 2017.

[37] R. Petroccia, J. Śliwka, A. Grati, V. Grandi, P. Guerrini, A. Munafò, M. Stipanov, J. Alves, and R. Been, "Deployment of a Persistent Underwater Acoustic Sensor Network: The CommsNet17 Experience," in *Proceedings of MTS/IEEE OCEANS 2018*, Kobe, Japan, May 28–31 2018.

[38] SECO, "Seco embedded board," http://www.seco.com/prods/it/form-factor/pico-itx-10x72cm/sbc-992-pitx.html, Last time accessed: March 2018.

# Document Data Sheet

| | | |
|---|---|---|
| *Security Classification* | | *Project No.* |
| *Document Serial No.* CMRE-PR-2019-024 | *Date of Issue* May 2019 | *Total Pages* 10 pp. |

*Author(s)*

Roberto Petroccia, Giovanni Zappa, Thomas Furfaro, Joao Alves and Luigi D'Amaro

*Title*

Development of a Software-Defined and Cognitive Communications Architecture at CMRE

*Abstract*

This paper presents an overview of the functionalities of the Cognitive Communications Architecture (CCA) that is currently under development at the NATO Science and Technology Organisation (STO) Centre for Maritime Research and Experimentation (CMRE). The CCA is designed to enable the deployment of advanced autonomous underwater solutions making use of smart, adaptive and secure underwater networking strategies. The CCA and the implemented network modules have been extensively tested, validated and improved during various at-sea campaigns conducted in recent years, involving CMRE and partners. The collected results show that the CCA is a robust, reliable and effective solution supporting underwater communications and networking. It provides various functions and services for the development of novel cognitive and secure communication strategies, such as a cross-layer networking functionality and the ability to easily integrate various existing communications technologies. These aspects are key enablers in the construction of a system that is robust to challenging conditions, such as those posed by varying channel conditions or adversarial attacks.

*Keywords*

*Issuing Organization*

NATO Science and Technology Organization
Centre for Maritime Research and Experimentation

Viale San Bartolomeo 400, 19126 La Spezia, Italy

*[From N. America:*
STO CMRE
*Unit 31318, Box 19, APO AE 09613-1318]*

Tel: +39 0187 527 361
Fax:+39 0187 527 700

E-mail: library@cmre.nato.int