Reprint Series CMRE-PR-2019-116

NEMO: An architecture for software communications research in the maritime domain

Arjan Vermeij, Thomas C. Furfaro, João Alves

June 2019

Originally published in:

Proceedings of the OCEANS 2015 MTS/IEEE Conference, 18-21 May 2015, Genoa, Italy, doi: 10.1109/OCEANS-Genova.2015.7271694

About CMRE

The Centre for Maritime Research and Experimentation (CMRE) is a world-class NATO scientific research and experimentation facility located in La Spezia, Italy.

The CMRE was established by the North Atlantic Council on 1 July 2012 as part of the NATO Science & Technology Organization. The CMRE and its predecessors have served NATO for over 50 years as the SACLANT Anti-Submarine Warfare Centre, SACLANT Undersea Research Centre, NATO Undersea Research Centre (NURC) and now as part of the Science & Technology Organization.

CMRE conducts state-of-the-art scientific research and experimentation ranging from concept development to prototype demonstration in an operational environment and has produced leaders in ocean science, modelling and simulation, acoustics and other disciplines, as well as producing critical results and understanding that have been built into the operational concepts of NATO and the nations.

CMRE conducts hands-on scientific and engineering research for the direct benefit of its NATO Customers. It operates two research vessels that enable science and technology solutions to be explored and exploited at sea. The largest of these vessels, the NRV Alliance, is a global class vessel that is acoustically extremely quiet.

CMRE is a leading example of enabling nations to work more effectively and efficiently together by prioritizing national needs, focusing on research and technology challenges, both in and out of the maritime environment, through the collective Power of its world-class scientists, engineers, and specialized laboratories in collaboration with the many partners in and out of the scientific domain.

NEMO: An Architecture for Software Communications Research in the Maritime Domain

Arjan Vermeij, Thomas C. Furfaro, João Alves
NATO STO Centre for Maritime Research and Experimentation
La Spezia, Italy
{Arjan.Vermeij,Thomas.Furfaro,Joao.Alves}@cmre.nato.int

***Abstract*—This paper presents a high-level overview of NEMO, an implementation of a communications research software workspace with special focus on application scenarios involving underwater (e.g. acoustic), and other maritime communications methods. The NEMO is an organic result of marine autonomy and communications experimentation, basically being composed of reconfigurable modules and services that interact with each other via an IPC (MOOS). The nominal NEMO configuration includes drivers for hardware devices, a simple MAC scheme, encoding, bundling, and queuing utilities, as well as a simple flooding router. Additionally, an implicit acknowledgement algorithm is proposed, the *herald*, that provides a mechanism for arbitrary payloads to be synchronised across the network with a configurable level of assurance. Lastly, an abstraction called the codex is presented, providing queuing, bundling and encoding functionalities in a single process, providing a high-level, packet-agnostic staging area for application data.**

***Index Terms*—underwater technology, underwater communications, underwater acoustics, underwater modem, communication system software, middleware, networking,**

I. Introduction

This paper presents the NEMO a prototype communications architecture with special focus on applicability in the underwater and maritime domains. The NEMO is the current incarnation of the communications system employed by the CMRE, used in Anti-Submarine Warfare (ASW), Mine Counter Measures (MCM), the underwater communications project, and other areas of marine autonomy research. Underwater communication technologies, beyond the well-enumerated physical limitations ([1]), exist in a relatively small, expensive market that significantly lags behind its terrestrial analogue. This in turn gives rise to a marked lack of standardisation and interoperability among underwater communications systems, a deficit that the CMRE is heavily invested in mitigating. The NEMO is a tool, a part of the framework in which standardisation and interoperability problems are being addressed, and as such a prototype, evolving constantly as the goals and requirements become more well understood. The remainder of the text presents an abbreviated history of the evolution and development of the NEMO in the context of other CMRE research endeavours, the current NEMO architecture and capabilities, and concludes by presenting the direction of future developments.

II. History & Context

At the CMRE, the primary driver of underwater communications development has been a requirement for data exchanges between mobile, potentially autonomous nodes, including cooperative (UxV to UxV) and command and control (C2) exchanges, in a mixed-media (aerial and submerged) environment. As such, the NEMO evolution has been emergent in nature, deriving from many related projects and efforts.

A. pAcommsHandler

One of the original modems used in CMRE autonomy experiments was the WHOI Micro-Modem ([2]), commonly used in research areas due to its open nature and relatively low cost. The software package initially used to operate the Micro-Modem is called `pAcommsHandler` ([3], [4]), originally developed at the Massachusetts Institute of Technology, and now maintained at GobySoft, LLC ([5]). The `pAcommsHandler` process is designed to interface with the MOOS ([6]) middleware, and includes a modem-specific driver, a MAC scheme, a queuing system, and an encoding mechanism (Dynamic Compact Control Language, DCCL, [7]). The `pAcommsHandler` and it's core functions can be configured at run time, using XML documents to describe the configuration.

While `pAcommsHandler` proved to be useful, in particular when operating on a Micro-Modem based network, the process was difficult to adapt to other modem types. Additionally, `pAcommsHandler` did not provide any routing facilities, and inserting additional "layers" between the existing core components was deemed to be a poor design decision. End users required higher data rates and increased range, indicating that a different modem was needed.

B. Software-Defined Open Architecture Modem

The Software-Defined, Open Architecture Modem (SDOAM), presented in [8], [9], is a Centre-driven design for an extensible, mixed-media communications software architecture specification for enabling inter-domain information exchange, with special, but not exclusive focus on underwater acoustic applications and bridging under- and above-water domains. In particular, the SDOAM concept strives for the capability to mix originator-heterogeneous software modules and capabilities to build complex communications system configurations. A key design feature of the SDOAM is the ability for proprietary, licensed,

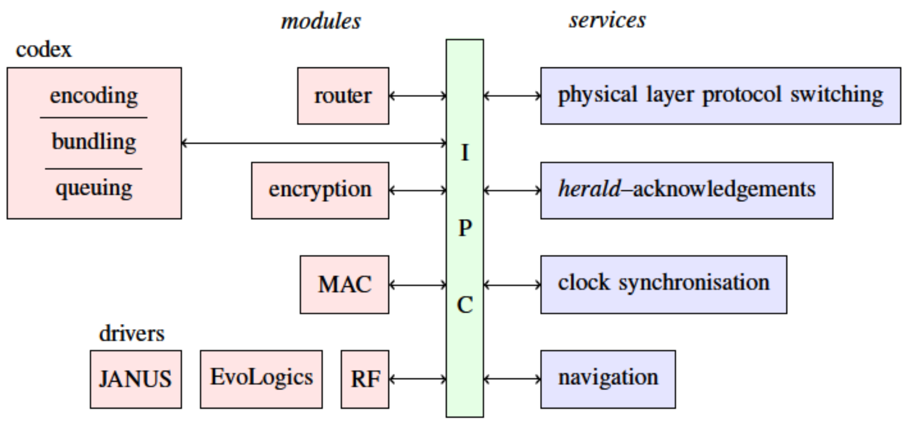

Fig. 1. NEMO system diagram.

and/or closed-source modules to be deployed in a way that ensures intellectual property preservation. The SDOAM design effort has largely been top-down, with focus on high-level architectural requirements. The NEMO presented in this report is a development implementation of the SDOAM in which to refine concepts and requirements of the SDOAM definition, via deployment of real communications system solutions in relevant operating scenarios.

C. *JANUS*

JANUS, presented in [10], is a modulation and coding scheme developed at the Centre in open coordination with industrial and academic partners, with the goal of providing a standardised, common language for underwater acoustic devices. JANUS is currently staged to become a STANAG, with convergence to the final specification ongoing. JANUS and the current NEMO design are orthogonal components of the future SDOAM – JANUS focuses almost wholly on physical layer concerns of acoustic communications, whereas the NEMO is the implementation of a high-level communications architecture that provides a workspace where JANUS can fit with other components.

III. NEMO

The NEMO is an effort to step away from the inherited legacy of `pAcommsHandler` (and in particular the non-CMRE-owned components) towards a more flexible framework for maritime communications research. The concept, depicted in Fig. 1, is to maintain the use of an IPC backbone for communication between related *services* and *modules*. Modules are processes that contribute directly to the transmission of data, while services are processes that provide functionality that is orthogonal to the data-transmission functionality. Unlike other OSI-like layouts, the NEMO explicitly does not maintain a sense of *layers* – any NEMO configuration formulation may be organised into layers, but it is not an express requirement. This is particularly important in the design and development of algorithms that require information from 'upper' layers, which breaks the layered paradigm.

NEMO attempts to address the limitations of the `pAcommsHandler` design and accomplish the task of interfacing with new hardware. The current NEMO implementation draws strongly on some of the `pAcommsHandler` modules, using the DCCL, and queue libraries in compile-time independent MOOS processes. The first iterations principally replaced the Micro-Modem driver with another hardware specific driver existing in a separate MOOS process, and connecting to the modem hardware via an interface process (either TCP/IP or serial). Additionally, routing was introduced via a simple flooding router, called `fladder`, which simply repeats every received packet for nearby nodes to hear. The process `sui-generis` is a complement to the `fladder` process, filtering duplicate packets from being transmitted or received. Fig. 2 depicts NEMO in one of it's earliest configurations.

One of the key features of NEMO, arising largely from the flexibility of the underlying IPC, is the run-time reconfigurability. As will be shown, creative use of the `fladder` and `sui-generis` processes with additional hardware drivers easily enables multimedia communications.

A. *Multi-media Communications*

A common topological concept in hybrid under- and above-water configurations is that of a *gateway*. A gateway is a node that has expressions in multiple communications channels, such as acoustic and RF. The NEMO can be configured in a way to provide such a gateway functionality, such as shown in Fig. 3. During the REP14–Atlantic cruise in July 2014, each node that had both RF and acoustic connections was configured to have a flooding router (`router.fladder`) between the on-board application (via `dccl`) and each physical channel, and also between the RF and acoustic channels. In effect, any packet that was heard from any information source (be it another node via RF or acoustics, or the local node from the application segment) would be retransmitted to all the other channels, subject to the filtering of the `router.sui-generis` process.

This bridging configuration of the NEMO is notable, in that it shows how the NEMO is mostly a workspace where

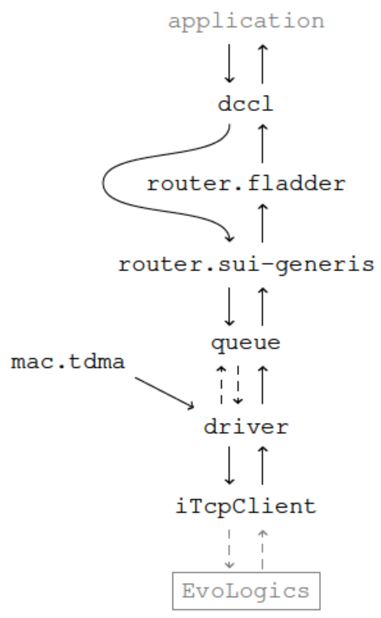

Fig. 2. NEMO baseline architecture. The processes `dccl`, `queue` are derivatives of the `libdccl` and `libqueue` of the original `pAccomsHandler`. All connections represented by solid lines are message-based interfaces passed via the IPC (MOOS), while the dashed lines represent TCP/IP client/server connections.

different processes can be quickly configured to have different behaviours, without recompiling code.

B. On-line relative clock synchronisation

The relative synchronisation of nodes without specialised hardware solutions is still a relatively open question for the underwater domain, [11], [12]. At minimum, *relative* synchronisation is an important capability for submerged systems, for data correlation between submerged systems, and relative navigation applications. In [12], a distributed synchronisation routine is presented, with at-sea data, collected using the NEMO.

C. Herald – Distributed Synchronisation

Acknowledgements are a prevalent concept in communications and distributed algorithm design. They are required when a data originator must have knowledge of the data consumer receiving successfully. When the underlying link between nodes is unreliable, acknowledgements simultaneously become more necessary (to ensure a level of application-to-application connectivity) and more cumbersome (because of unreliability and induced overhead).

The distinction between *explicit* and *implicit* acknowledgements are made, such that explicit acknowledgements are asynchronously generated at the data sink and implicit acknowledgements are all other forms, i.e., synchronous replies that may be part of other modem packets. In a medium with a very high propagation delay, explicit acknowledgements impose a high penalty on effective bandwidth, and are generally avoided in networked underwater scenarios (i.e., not solely point-to-point communications). An implicit acknowledgement service is proposed, as an alternative to the classic explicit acknowledgement scheme, called the `herald`.

The `herald` process tries to ensure that some piece of information, called the *payload*, is equal on a number of participating nodes. In order to do so `herald` maintains a status bit for each participating node. A value of 1 means the corresponding participating node is known to have or to once have had the same payload as on the originator. A value of 0 means the corresponding participating node is known not to have the same payload payload, or is not known to have the same payload as we have. The herald also maintains a counter of the current to-be-heralded payload, to be able to understand whether an incoming, different payload is older or newer than the current payload. The status bits and the payload counter are together called the *header*, and are combined with the payload into an *envelope* when communicating with other nodes.

Whenever the payload changes, `herald` transmits the envelope to all participating nodes. On each node that receives the envelope, the `herald` will accept the new payload, counter, and status bits, set its own status bit to 1, and re-send the envelope to let other participating nodes know that it now has the new version of the payload. If `herald` receives an envelope with a payload and counter it already has, it compares the received status bits with its own status bits. If these are different, it will merge the received status bits with its own status bits and re-send the envelope. Finally, if a node receives an envelope, no matter what the content, from a node for which the status bit is 0, `herald` will re-send the envelope. This mechanism will converge on the state where every participating node has the same payload and all status bits are set to 1 on all nodes. At this point this, all nodes have acknowledgement that all other nodes have received this specific payload.

Additionally, there are cases where the status flag for the heralded payload may need to have multiple states. In particular, some applications using the herald to distribute the payload information may require the nodes not only receive the payload, but have acknowledgement that other nodes have the payload as well. An example is the case of ad-hoc networking with nodes inserting themselves into an existing MAC scheme, like TDMA. Once a new node arrives in the network, a new schedule must be generated to accommodate the new node, and this insertion must be communicated to all nodes in the network. When the schedules have been updated globally, only then should the TDMA associate with the new schedule, after all nodes are explicitly aware that all other nodes are ready to make the change (i.e., they have updated their local copy of the schedule). In other words, every node ”needs to know that all the nodes know that all nodes know” about the new information. As such, the conception of the herald does not preclude multi-state status flags.

The herald is meant to be a service for other modules, who may request that their application specific payload be synchronised across the network. For example, in a network where all nodes have multiple acoustic modulation and coding

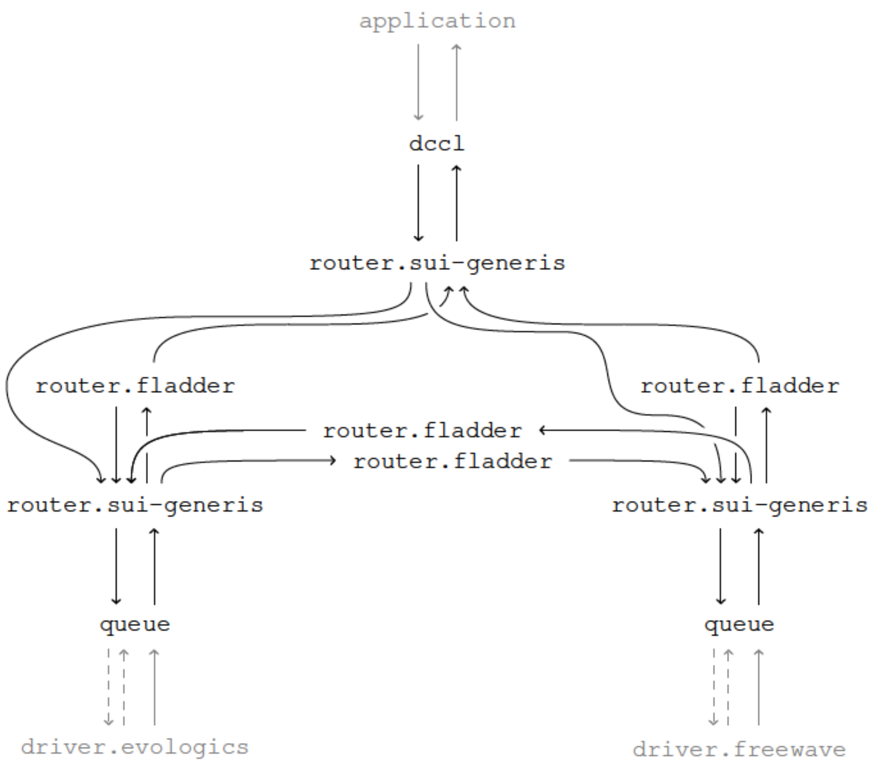

Fig. 3. The REP14–Atlantic routing architecture. `driver.evologics` and `driver.freewave` are interfaces to acoustic and RF modem hardware, respectively. Instances of the flooding router `router.fladder` and router filter `router.sui-generis` are placed between all three of the local applications, and each media endpoint.

capabilities, such as JANUS mixed with other proprietary types, the herald could be used to distribute the knowledge of each nodes capabilities. The herald is under active development, having been tested in simulation and scheduled for in-water testing in 2015.

D. Codex – Queuing, Encoding and Bundling

Applications typically deployed alongside underwater communications systems are highly coupled to the unique characteristics of acoustic communications, in particular the high latency, low bandwidth, and transmission cycle period. Such coupling severely limits the adaptability of applications to different communications infrastructures (even different modems), and moreover burdens the application developers to be conscious of the underlying transport mechanism. This is fundamentally different than classical terrestrial communications.

As an alternative to the "packet-centric" paradigm, here the *codex* is proposed. This codex is intended to replace the current queue, bundling, and encoding functions of the `pAcommsHandler` queue and DCCL. The codex is the aggregate of all data that must be shared with others nodes in the network, and is maintained by the `composer` process. Applications can add new information to the codex, replace existing information with more up-to-date information, and remove information which is no longer deemed relevant. The codex is structured hierarchically in a tree of *segments*, *sets*, and *fields*.

The terms *segment*, *set*, and *field* are borrowed from the ADatP-3 (Allied Data Processing Publication number 3) [13] standard. The codex structure is described in an XML schema, with specific codex instances being contained in an XML document.

a) Fields: A field is the atomic data element, that is, a piece of data that cannot be decomposed further into an array or hierarchy.

b) Sets: A set is the smallest piece of information that is self-comprehensible, without any other context. A set consists of fields, with each field only making sense if received with all other fields of the same set.

c) Segments: A segment adds hierarchy to the codex. A segment may contain sets as well as other segments. The codex is itself a segment.

1) Bundling: When a time slot for transmission begins, it is the task of the composer to encode the codex and send it out. In general the codex will be too large to fit into a single modem packet (e.g., EvoLogics modem packets are 64 B maximum). The composer distributes the codex over several modem packets, where each modem packet contains a *partial codex*. Since it's not likely all modem packets will arrive at their destination, care must be taken that the receiver has all the information required to recreate the partial codex contained in each packet. In Delay Tolerant Networking (DTN) this is known as the *Bundle Protocol* ([14], [15]).

For example, if the codex contains a position, it may be senseless to put the longitude in one modem packet, and the latitude in another. For this purpose the composer guarantees that the contents of one set will never be distributed over more than one modem packet.

In other cases a set can only be interpreted correctly with information contained in another set, called a *context set*. The composer only inserts a set into a modem packet if all its context sets can be inserted as well. Conversely, the composer will not insert a set into a modem packet if it is a context set of some other set and that other set cannot be inserted as well.

It's of course possible that a set, or the combination of a set and its context sets, is too large for encoding into one modem packet. Obviously such sets will never be selected for transmission via the modem. It may be that other media do accept larger packets, sets that are too big to be sent via the modem may be sent via these other media, if and when available. It's currently up to the designer of the codex to guarantee that sets are small enough to be sent out via the intended media. This is not to say that the application-level data structure (of whatever type) that an application encodes is required to be smaller than a packet or even an entire frame, but that the application is responsible for the division of the data into appropriate sets.

2) Encoding: Encoding is the process of converting part of the codex to a modem packet, decoding is the reverse. The primary aim is to get as high a data compression ratio as possible. For a modem packet to be decodable, it needs to contain the values of all fields of the encoded partial codex, as well as structural information needed to understand which fields of which sets are contained in the modem packet. Moreover, the transmitter and the recipient of a modem packet need to have the **exact same copy** of the codex description.

a) Fields: Different types of fields require different compression methods.

boolean

Boolean fields can have only one of two values, *false* and *true*. The encoding is straightforward: 0 for *false*, 1 for *true*. The number of required bits is exactly 1.

enumeration

An enumeration is a set of labels $l_0, l_1, \ldots, l_i, \ldots, l_{N-1}$. An enumeration field can have only one of the N values of the set. Instead of encoding the label itself, it suffices to encode the index i of the label, where $0 \leq i < N$. The number of required bits is equal to $\log_2(N)$.

fixed point

A fixed-point number is a number with a precision that is constant over the entire domain. In general a fixed-point number can have any value from the domain of real numbers $\mathbb{R}$. For the purpose of encoding the domain needs to be constrained by supplying lower and upper bounds L and U respectively. Optionally a *granularity* g can be specified, g defaults to 1. The granularity equals the precision. The value of a fixed-point field is constrained to the set $L, L+g, \ldots, L+ig, \ldots, L+(N-1)\,g, U$. For the purpose of encoding it suffices to encode the index i. If $(U-L) \equiv 0 \pmod g$, then $L+(N-1)\,g = U$ and the number of required bits equals $\log_2(N)$. Otherwise, if $(U-L) \not\equiv 0 \pmod g$, then the number of required bits equals $\log_2(N+1)$.

floating point

A floating point number, v, is represented by a *significand* s, a base b, and an exponent e, where $v = s \times b^e$, where the base b is set to 10. The values for s and e are themselves fixed-point numbers with a granularity of 1. Encoding a floating point number is therefore the composition of the encoding of the two fixed-point numbers s and e.

3) Relative encoding: In some cases it may be advantageous to not send the value of a field, but the difference with respect to the value of another field. For example, a modem time stamp is an integer number between 0 and $16 \times 2^{32} - 1 = 68719476735$, in microseconds. The granularity is 100 microseconds, the number of bits required is $\log_2\left(16 \times 2^{32}/100\right) \approx 29.4$[1] If we send more than one modem time stamp, and we restrict the difference between the oldest and the newest time stamp to 5 minutes, then the number of bits required for a relatively encoded time stamp value is $\log_2\left(5 \times 60 \times 10^6/100\right) \approx 21.5$, a reduction of 27%.

4) Structure: A modem packet contains a partial codex. The encoding process must include enough information such that the recipient is able to understand which part of the codex, or which sets and segments, a modem packet contains. For this purpose a *flag* is inserted before each set and segment in the tree. The flag tells the recipient whether or not the next set or segment in the tree is present. Thus the receiving `composer` process can traverse the packet as it decodes the partial codex.

E. Queuing

The current queue mechanism is that of `pAcommsHandler`, a prioritisation mechanism is based on application messages that have a time-to-live (TTL) and priority. The codex instead bases the prioritisation on the included sets. Additionally, the codex queueing system is different than that of `pAcommsHandler` as it provides a mechanism for the order of outgoing partial codexs to be explicitly configured.

F. Repeated transmissions

In some cases it may be desirable to transmit the same information more than once, possibly within the same time slot. A repeated transmission has a higher probability to reach its destination. To accommodate for repeated transmissions, a *repeat* attribute can be added to a set.

[1]The decimal number of bits indicates the exact number of bits required to represent the states in the range. Rounding the bits immediately would incur some small inefficiency in the compression, where the field might be combined with other fields in a way that utilises the full range of values more efficiently.

G. Development

The codex concept is subject of ongoing refinement and preliminary implementation. First steps will include simulation and modelling of NEMO while using the codex, to quantify the efficiency and performance with respect to the previous `pAcommsHandler` implementation.

IV. Conclusion

This paper describes the NEMO, a communications system and architecture that is composed of a set of services and modules connected via an IPC (MOOS, currently). The NEMO is intended to be a workspace or framework where communications-centric processes and algorithms can be tested, with emphasis on reconfigurability and explicit lack of formal structure. The NEMO is the current implementation of the maritime-centric communications stack used at CMRE for underwater communications and autonomy research, providing a space where efforts such as JANUS, the SDOAM, clock-synchronisation, and routing can be deployed and tested. Finally, the ongoing development of an acknowledgements-as-a-service algorithm (the `herald`) and a modem-packet-agnostic queuing, bundling and encoding mechanism, the codex, are presented.

Acknowledgement

This work has been partially funded by the European Unions Seventh Framework Programme for research, technological development and demonstration under grant agreements no. 288704.

References

[1] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communication and networking: Recent advances and future challenges," *The Spring 2008 MTS Journal, The State of Technology in 2008*, vol. 42, no. 1, pp. 103–116, 2008.

[2] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The whoi micro-modem: an acoustic communications and navigation system for multiple platforms," in *OCEANS, 2005. Proceedings of MTS/IEEE.* IEEE, 2005, pp. 1086–1092.

[3] T. Schneider and H. Schmidt, "Goby-acomms: A modular acoustic networking framework for short-range marine vehicle communications," *URL http://gobysoft. com/dl/goby-acomms1. pdf*, 2012.

[4] T. E. Schneider and H. Schmidt, *MOOS-IvP Communications Software with acoustic networking from the Goby Underwater Autonomy Project*, Aug. 2010.

[5] T. E. Schneider, *Goby Underwater Autonomy Project*, Feb. 2011.

[6] M. R. Benjamin, P. M. Newman, H. Schmidt, and J. J. Leonard, "A tour of moos-ivp autonomy software modules." MIT Computer Science and Artificial Intelligence Lab, Tech. Rep. MIT-CSAIL-TR-2009-006, Jan. 2009.

[7] T. Schneider and H. Schmidt, "The dynamic compact control language: A compact marshalling scheme for acoustic communications," in *Oceans 2010 IEEE-Sydney*. IEEE, 2010, pp. 1–10.

[8] J. Potter, J. Alves, T. Furfaro, A. Vermeij, N. Jourden, G. Zappa, A. Berni, and D. Merani, *Software Defined Open Architecture Modem Development at CMRE, CMRE-FR-2013-023*, Dec. 2013, NATO UNCLASSIFIED.

[9] J. Potter, J. Alves, T. Furfaro, A. Vermeij, N. Jourden, D. Merani, G. Zappa, and A. Berni, "Software defined open architecture modem development at CMRE," in *Underwater Communications and Networking (UComms), 2014*. IEEE, 2014, pp. 1–4.

[10] J. Potter, J. Alves, D. Green, G. Zappa, K. McCoy, and I. Nissen, "The JANUS underwater communications standard," in *Underwater Communications and Networking Conference (UComms), Sestri Levante, Italy*, Sep. 2014.

[11] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2005.01.002

[12] A. Vermeij and A. Munafo, "Clock synchronisation in underwater acoustic networks," in *Underwater Communications and Networking (UComms), 2014*. IEEE, 2014, pp. 1–5.

[13] *ADatP-3(A) NATO message text formatting system (FORMETS), concept of FORMETS (CONFORMETS)*, Change 1 ed., NATO Standardization Agency, Brussels, 2009, nATO/EAPC UNCLASSIFIED, Releasable to Australia, MD, ICI.

[14] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. (2007, Apr.) Delay-tolerant networking architecture, irtf dtn research group. IETF RFC 4838. [Online]. Available: http://tools.ietf.org/html/rfc4838

[15] K. Scott and S. Burleigh. (2007, Nov.) Bundle protocol specification. IETF RFC 4838. [Online]. Available: http://tools.ietf.org/html/rfc5050

Document Data Sheet

Security Classification		*Project No.*
Document Serial No. CMRE-PR-2019-116	*Date of Issue* June 2019	*Total Pages* 6 pp.

Author(s)

Arjan Vermeij, Thomas C. Furfaro, João Alves

Title

NEMO: An architecture for software communications research in the maritime domain

Abstract

This paper presents a high-level overview of NEMO, an implementation of a communications research software workspace with special focus on application scenarios involving underwater (e.g. acoustic), and other maritime communications methods. The NEMO is an organic result of marine autonomy and communications experimentation, basically being composed of reconfigurable modules and services that interact with each other via an IPC (MOOS). The nominal NEMO configuration includes drivers for hardware devices, a simple MAC scheme, encoding, bundling, and queuing utilities, as well as a simple flooding router. Additionally, an implicit acknowledgement algorithm is proposed, the herald, that provides a mechanism for arbitrary payloads to be synchronised across the network with a configurable level of assurance. Lastly, an abstraction called the codex is presented, providing queuing, bundling and encoding functionalities in a single process, providing a high-level, packet-agnostic staging area for application data.

Keywords

Underwater technology, underwater communications, underwater acoustics, underwater modem, communication system software, middleware, networking

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