A JANUS-Based Consensus Protocol for Parametric Modulation Schemes

Emil Wengle*
Norwegian University of Science and Technology (NTNU)
Department of Electronic Systems
Trondheim, Norway
emil.wengle@ntnu.no
un.identified.nu@gmail.com

John Potter
Norwegian University of Science and Technology (NTNU)
Department of Electronic Systems
Trondheim, Norway
john.r.potter@ntnu.no

Damiano Varagnolo
Norwegian University of Science and Technology (NTNU)
Department of Engineering Cybernetics
Trondheim, Norway
damiano.varagnolo@ntnu.no

Hefeng Dong
Norwegian University of Science and Technology (NTNU)
Department of Electronic Systems
Trondheim, Norway
hefeng.dong@ntnu.no

ABSTRACT

JANUS, the first digital underwater acoustic communications standard, is robust to noise in low SNR situations, but offers very limited data rate. A primary use of JANUS is thus that of a first-contact language in underwater networking, and may be laddered by a network as a means to decide in a distributed and collaborative manner which communication scheme and modulation parameters should be used after the first contact. Deciding this distributedly implies then negotiating a common choice, likely suboptimal for the individual node, but better holistically.

This paper proposes, mathematically analyses and tests in simulation such a distributed modulation parameters negotiation protocol based on an ad-hoc consensus paradigm. In this, the nodes start with already estimated locally optimal modulation parameters (an estimation step outside the scope of this paper), and then exchange opportune series of opportunely crafted JANUS messages that bring nodes to a global consensus on such parameters. Besides proposing this protocol, we describe a necessary and sufficient convergence criterion under the assumption that the channel coherence time is large. In practice, we show that convergence is ensured as soon as the coherence time of the link reliability is much longer than the convergence time for the proposed negotiation process.

We test, using simulations implemented in UnetStack, how often the protocol successfully achieves consensus in other realistic situations, and find that convergence may be achieved with a consensus error rate on the order of one per thousand in a three-node network, and an error rate on the order of one percent in a seven-node network with weak links.

*Please send any correspondence to emil.wengle@ntnu.no – the other address is there solely for technical reasons.

WUWNet'22, November 14–16, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9952-4/22/11.
https://doi.org/10.1145/3567600.3568142

CCS CONCEPTS

• **Networks** → **Network protocol design**; • **Computing methodologies** → *Distributed algorithms*; • **Mathematics of computing** → Network flows.

KEYWORDS

JANUS, consensus algorithms, underwater acoustic networks

ACM Reference Format:
Emil Wengle, John Potter, Damiano Varagnolo, and Hefeng Dong. 2022. A JANUS-Based Consensus Protocol for Parametric Modulation Schemes. In *The 16th International Conference on Underwater Networks & Systems (WUWNet'22), November 14–16, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3567600.3568142

1 INTRODUCTION

The past few decades have seen a growing demand for underwater acoustic sensor networks. With an ever-increasing range of applications, from offshore site monitoring to natural disaster alert systems and naval traffic surveillance, it has not come as a surprise that the vision of the Internet of Underwater Things has been proclaimed by researchers in the field [5]. There exist many standards in terrestrial wireless radio communication. The Internet of Things builds upon a wide variety of wireless communication standards, including, but not limited to, ZigBee and IEEE 802.11ah [15]. Until recently, however, there were no standards for digital underwater acoustic communication. The only such standard that exists is ANEP-87, also known as JANUS [14]. It is robust by design, but offers a data rate of only 80 bits/s in its original frequency band. While this enables reliable communication over several kilometres, JANUS achieves far lower throughput than offered by many commercially available underwater acoustic modems [9].

The flexible packet definition of JANUS encourages a wide range of applications, from making first contact to sending distress messages [6, 12]. The idea of using JANUS to switch languages was first implemented in the work of [13], where the authors used a higher-frequency adaptation of JANUS for making first contact and switching languages on Applicon Sea-Modems. The authors of [1] saw the need for standardising the mechanism for discovering the network and selecting a language for communication. Numerous

efforts have been made to this end. For example, [12] demonstrated experimentally the case of using JANUS to make first contact with other nodes in a network and discovering their capabilities. However, the proposed application considered only existing proprietary communication schemes, and the scheme selection strategy was basic. The authors thus saw the need for a more robust and richer selection mechanism. Recently, [11] proposed a first contact protocol, built on JANUS, that assigns identifiers to agents that wish to join the network, such that they are all unique within a two-hop neighbourhood. While this protocol is more sophisticated than that in [12], the author left the language switching process for future work, because all agents in the network supported the same scheme.

Statement of contributions. To this end, we consider a consensus-based approach to selecting parameters in a modulation and coding scheme. First, the nodes exchange opinions on channel properties and reach consensus on them. Then, the nodes find suitable orthogonal frequency division multiplexing (OFDM) parameters in a distributed way.

We then specifically focus on understanding how to use JANUS to execute the first part of this consensus procedure. In this way we let the parameter selection be a generic consensus scheme, and instead focus on devising a solution that is device agnostic, following the paradigm shift towards software-defined, open-architecture modems [4].

We note that we assume static network topologies with unreliable links, since network topologies with mobile nodes are left for an extension work. Solving the challenges associated to this situation requires dealing with more details than the ones that may be described in a pages-constrained manuscript.

2 PROTOCOL

This section describes the proposed JANUS-based protocol for agreeing which OFDM parameters should be used by the network after the first encounters phase. Instrumental to this, we summarise in Table 1 and in the next subsection the bit assignment in the JANUS baseline packet. Numbers (unsigned) that are given are fixed for that field. A dash indicates that any number that the allocated bits allow may be used.

2.1 Types and structure of the packets

The proposed protocol uses JANUS packets formed as in Table 1 to indicate two different types of operations: one for communicating channel estimates to the peers, and another for requesting the consensus process to be continued.

The information relative to the type of operation and the to-be-exchanged channel estimates needs to be encoded on bits slots that are not being used by JANUS. Our proposal (see again Table 1) is to encode it within JANUS' open class user ID and following application types bits slots.

More specifically, as for the first type of to-be-sent message (i.e., the *channel estimate* one) we let it be indicated by an application type 2, and to contain the 8-bit identifier of the transmitting node, a pair of bits that together define the subclass of the message, and 20 bits that together quantify the channel in terms of delay spread, Doppler spread and noise level. We recall that the focus of the paper is on the consensus protocol, and not on how to estimate the initial local opinions on which OFDM parameters should be used – something that we here assume as given.

Table 1: Bit allocation plan of the JANUS packets.

Field	Bits	Value
JANUS version	1–4	3
Mobility flag	5	—
Schedule flag	6	0
TX/RX flag	7	1
Forwarding capability	8	—
Class user ID	9–16	66
Application type	17–22	either 1 or 2
Application data block	23–56	as in Figures 1 or 2
8-bit CRC	57–64	automatically generated

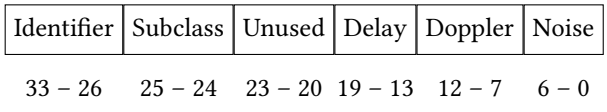

Figure 1: Structure of the 34-bits long *channel estimate* application data block (as indicated by setting the *Application type* bits to 2).

Figure 1 summarises the bit allocation plan for the application data block, as specified in [14]. Bit 33 in Figure 1 maps to bit 23 and bit 0 maps to bit 56 in Table 1. The delay spread has range 0 ms to 127 ms in increments of 1 ms. Doppler spread is also an unsigned number, ranging from 0 Hz to 31.5 Hz in steps of 0.5 Hz. Finally, the noise level has range 32 dB re. 1 μPa to 159 dB re. 1 μPa in 1 dB increments.

As for the second type of to-be-exchanged message, we denote it as a *new round* packet that shall be sent by a node when it wants the consensus process to be continued. As will hopefully be clear below, such requests may arise when a node senses it is losing too many packets, or other conditions specific to the protocol we are proposing in this paper.

The information to be encoded in *new round* packets comprises two parts serving two distinct purposes: first, a "forcing" flag that instructs the receiving nodes to interrupt the running consensus process and start a new one. Second, an integer identifier of such interrupt, useful to determine whether one has already received that specific interrupt or not. Figure 2 shows the proposed bit allocation of the ADB of the packet.

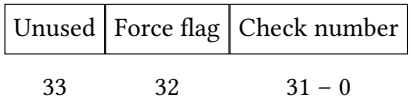

Figure 2: Structure of the 34-bits long *new round* application data block (as indicated by setting the *Application type* bits to 1).

2.2 Structure of the communication protocol

The proposed protocol is in short a series of rules stating when and under which conditions nodes should exchange *channel estimate* and *new round* packets. The protocol may thus be summarised as the finite state machine depicted in Figure 3. The following paragraphs describe in words this machine.

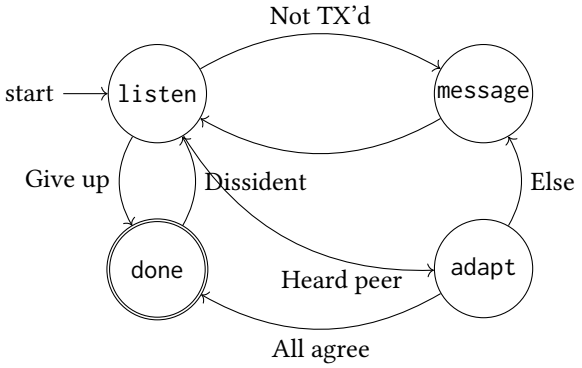

Figure 3: Finite state machine representation of the protocol.

More precisely, each node begins in the listen state, which they stay in for a random duration of $t = t_0 + T$ seconds with t_0 being the minimum waiting time and $T \sim \text{Exp}(\lambda)$ (with $\text{Exp}(\lambda)$ being the exponential distribution of density $p(x;\lambda) = \lambda \exp(-\lambda x)$, with λ a protocol parameter). The duration is generated every time it sets the listening timer, and it may be rounded to a time unit that the device supports, for instance milliseconds or microseconds, if necessary.

If a node receives a *channel estimate* message while in the listen state, it registers the opinion of the peer, stays in the listen state, and restarts its own listening state timer to keep listening for opinions.

If the listening timer expires while being in the listen state, instead, then the node checks which of the two conditions happened most recently: *a)* a transition from adapt to message, or *b)* a *new round* packet was sent or received.

- **If *a),*** the node enters the message state, transmits its opinion, and returns to the listen state. On entering, it temporarily increases the time spent listening to three times the normal time – an extra waiting time that accounts for the delay effects that the message has on other listeners as they reset their listening timers and increases the likelihood of hearing the opinion of a different node.
- **If *b),*** then the node checks if it has previously registered any other opinion from its peers. If the node has registered at least one opinion, it enters the adapt state, where it will do the operations summarised in the next paragraph. If it has not, and has not been to the message state three times since the start of the round or its last visit to the adapt state, it enters the message state, and behaves as described in the "*a)*" case. It sends a *channel estimate* message again to mitigate the risk of others ending the consensus process too soon. After attempting three times in a row, the node gives up and goes to the done state if it has not adapted this round, because it has reason to believe that there is nobody to communicate with. If it has adapted at least once this round, it goes to the adapt state anyway, because the node knows that there are others to communicate with, and the rest of the network could have reached consensus.

The message state employs a simple carrier-sensing medium access (CSMA) scheme to mitigate the risk of packets colliding in the network. Before the node transmits a *channel estimate* message, it checks if the channel is busy such that its transmission does not interfere with its own reception. If the channel is deemed to be busy, it waits for $0.05 \cdot 2^{N-1}$ seconds before it tries again, where N is the number of times it sensed a busy channel since the last time it sent a message. If it senses a busy channel $N_{\max}$ times in a row, the node abandons the attempt to transmit its message, and returns to the listen state. For the purposes of this paper, we let $N_{\max} = 8$ to make the CSMA scheme comparable to the one defined in the standard. As soon as it senses an available channel before that happens, it sends its message.

When the node enters the adapt state, the node first checks the opinions it has registered. If all registered opinions are equal to its own opinion, or if the register is empty, but the node has not given up, it recognises that it reached local consensus. This opinion is broadcast to the network, and the node enters the done state. Otherwise, it computes its next opinion as described in Section 2.3.

When a node enters the done state, unless it has given up, it computes its OFDM communication parameters from the channel properties it found from the consensus process. When those parameters have been found, the node sets a timer on a fixed duration. This duration may be selected freely, but should be set long enough to allow all nodes to enter the done state before the timer on the first node expires. The node may send and receive OFDM packets only while in this state.

A node may leave the done state in three ways. The first way is when the OFDM communication timer expires, at which point the node initiates a new round of consensus. When it does so, it broadcasts a non-forcing *new round* packet and enters the listen state. Nodes that receive a *new round* packet and also are in the done state forward the packet if they can, and reset their state machine. It needs not be in the done state to reset the state machine if the forcing flag is set in the packet; see the explanation to Figure 2.

The second way is when the node experiences difficulties with decoding OFDM packets. If it loses three OFDM packets in a row, it sends a forcing *new round* packet, as explained in Section 2.1, aiming to negotiate a channel state that results in better suited modulation parameters.

The third way is when the node overhears an opinion that differs from what the node believes is the local consensus, provided the OFDM timer has more than half the time remaining. When this happens, the current round is resumed in fine-tuning mode, because it is likely that the difference in the differing opinion to the node's own one is small. Until the node starts a new round, it skips the perturbation and rounds down the average it finds in the adaptation state. This transition is represented as the "dissident" edge in Figure 3.

2.3 Finding the next opinion

When a node enters the `adapt` state and has registered at least one differing opinion, it computes a compromise between the opinions that the node has registered since the last iteration. The node then updates its own opinion by taking a step with size $0 < \beta \leq 1$ towards the compromise opinion, rounding any values between two points such that the new opinion lands closer to the compromise. After rounding, the node goes to the `message` state to broadcast the new opinion. The opinions of others that the node had access to are forgotten when it exits the `adapt` state to ensure that only fresh opinions are used next time it enters the `adapt` state.

The compromise policy is user-defined, but the compromise should lie within or on the boundary of the convex hull of the opinions available to the policy. Just for demonstrative purposes, in this paper we consider a compromise that is found by averaging (in a quantised sense) the set of registered opinions with the node's own opinion. More details are given in Section 3.1.

3 CONVERGENCE ANALYSIS

The consensus problem is well known in the context of distributed control systems. The simplest variant of this problem considers stable and static network topologies and continuous decision variables, and there is a wide body of literature and papers on consensus algorithms and convergence of the consensus problem [3, 8, 10, 16]. We consider static network topologies, but with unreliable links to make our protocol applicable to realistic underwater sensor networks.

We represent a snapshot of the network at time t as the directed graph $\mathcal{G}_t = (V, E_t)$. $\mathcal{G}$ has $|V| = n$ nodes, and its edge set E_t is described by the adjacency matrix $\boldsymbol{A}_t$. Element $a_{ij,t}$ is one if edge $(i, j) \in E_t$ and zero otherwise. Because the links are unreliable, the adjacency matrix $\boldsymbol{A}_t$ can be thought of as a realisation of a matrix $\boldsymbol{B}$ of independent Bernoulli distributed random variables with probabilities $\boldsymbol{P} = E(\boldsymbol{B})$, such that $0 \leq p_{ij} \leq 1 \; \forall (i, j)$ indicate the reliability of the link from i to j.

At node i and time step t, the new opinion is calculated as

$$x_i[t+1] = \beta \frac{\sum_{j=1}^{n} x_j[t] a_{ji,t}}{\sum_{j=1}^{n} a_{ji,t}} + (1-\beta) x_i[t]. \tag{1}$$

Equation (1) is readily extended to the whole network with a few adjustments. Using Equation (1) as our update equation, we propose a criterion for the network to converge to consensus as $t \to \infty$ in Proposition 1.

Proposition 1. *Let $\bar{\mathcal{G}} = (V, \bar{E})$ be a weighted directed graph with adjacency matrix $\boldsymbol{P}$, such that $\bar{\mathcal{G}}$ models the expected structure of $\mathcal{G}$. Then, the network eventually converges to a common value for all initial state vectors $\boldsymbol{x}_0$ if and only if there exists a node $v \in V$, for which there exists a path to all other nodes $u \in V \cap \{v\}^C$.*

Before we begin the proof, we look back at our model. Equation (1) uses realisations of $\boldsymbol{P}$. However, finding the expected value of $\boldsymbol{x}[t+1]$ conditioned on $\boldsymbol{x}[t]$ is difficult due to the sum in the denominator. There is no closed-form expression of the expected value of the average of a sample of numbers, where each number is sampled with a given probability, not necessarily equal for each number. Fortunately, we only need to consider whether a link exists in our model; this is because $\boldsymbol{P}$ is constant, so the reliability does not change with time, and we are considering the steady-state solution $\boldsymbol{x}_\infty$. Less reliable links only delays convergence because packets are lost more often. We thus introduce $\boldsymbol{P}' = \operatorname{sgn} \boldsymbol{P}$, where $\operatorname{sgn} x$ is the sign function, and use $\boldsymbol{P}'$ in lieu of $\boldsymbol{P}$ to simplify the proof.

Letting $a_{ij,t} = p'_{ij}$ in (1) and $\beta = 1$[1] gives the deterministic update equation

$$x_i[t+1] = \frac{\sum_{j=1}^{n} x_j[t] p'_{ji}}{\sum_{j=1}^{n} p'_{ji}},$$

which, after stacking all x_i, gives

$$\boldsymbol{x}^T[t+1] = \boldsymbol{x}^T[t] \boldsymbol{U}, \tag{2}$$

where $\boldsymbol{U} = \boldsymbol{P}' \left(\operatorname{diag} \sum_{i=1}^{n} \boldsymbol{P}'_{i\cdot} \right)^{-1}$ is the update matrix, and $\boldsymbol{P}'_{i\cdot}$ denotes the i-th row of $\boldsymbol{P}'$. We also introduce the steady-state matrix

$$\boldsymbol{U}_\infty = \lim_{t \to \infty} \boldsymbol{U}^t$$

to aid us in our proof. It is idempotent, which means that $\boldsymbol{U}_\infty^2 = \boldsymbol{U}_\infty$.

We are now ready to give the proof to Proposition 1.

Proof. (The *only if* part)

We use proof by contradiction. Suppose that $\bar{\mathcal{G}}$ converges to a common value, but there does not exist a node, for which there exists a path to all other nodes.

Then, for all $v \in V$, there exists at least one node u that v does not have a path to. This means that the (v, u)-th element in the steady-state matrix $\boldsymbol{U}_\infty$ is zero, so $x_{u,\infty}$ is unaffected by $x_v[t]$ for all t. $\boldsymbol{x}_\infty = c\boldsymbol{1}$ is therefore not a steady-state solution for all $\boldsymbol{x}[0]$, and the network cannot converge to consensus, which is a contradiction. This concludes the proof of the *only if* part.

(The *if* part)

We also prove this part by contradiction. Suppose that there exists a node $v \in V$, such that for all $u \in V \cap \{v\}^C$, the path from v to u exists, but $\bar{\mathcal{G}}$ does not converge to a common value. Recall that the update matrix $\boldsymbol{U}$ is normalised such that its columns sum to unity, so $\boldsymbol{x}_\infty = c\boldsymbol{1}$ is a reachable steady-state solution to (2). We thus have to show that this is not the only possible solution in this case, or, equivalently, that the columns of the steady-state matrix $\boldsymbol{U}_\infty$ are not all identical. If they are all identical, then $\boldsymbol{x}_\infty^T = \boldsymbol{x}_0^T \boldsymbol{U}_\infty = c\boldsymbol{1}^T$ will be the only possible steady-state solution because $\boldsymbol{U}_\infty$ then has unit rank, and we have a contradiction.

To facilitate the proof, we assume that $\boldsymbol{U}$ is structured as

$$\begin{pmatrix} \boldsymbol{U}_{ss} & \boldsymbol{U}_{sk} \\ \boldsymbol{0} & \boldsymbol{U}_{kk} \end{pmatrix},$$

where $\boldsymbol{U}_{ss} \in \mathbb{R}^{m \times m}$ represents the effect of edges inside the source component, $\boldsymbol{U}_{sk} \in \mathbb{R}^{m \times (n-m)}$ the effect of those from the source to the rest of the network, and $\boldsymbol{U}_{kk} \in \mathbb{R}^{(n-m) \times (n-m)}$ the effect of edges inside the rest of the network.

Any node $u \in U \subseteq V \cap \{v\}^C$ that has a path to v forms a cycle with v, because v by definition has a path to all other nodes. Therefore, all nodes in U are also part of the source component. The cycle is closed to other nodes because they would otherwise be part of it, hence the zero block in the structure of $\boldsymbol{U}$. Consequently,

[1] Choosing $\beta < 1$ only slows convergence further, because it effectively adds more weight to the own state variable.

$u \in U \cup \{v\}$ will eventually converge to identical $x_{u,\infty}$ for all $\boldsymbol{x}_0$, which means the first m columns in $\boldsymbol{U}_\infty$ will be all identical.

Further, from the definition of v, there exists a $t' \leq |V| - 1$ such that the first m rows of $\boldsymbol{U}^{t'}$ contain all-nonzero elements. The other $n - m$ rows must converge to zero as $t \to \infty$ due to the structure of $\boldsymbol{U}$. Similarly, the column sum of the first m rows in $\boldsymbol{U}^t$ must approach $\mathbf{1}$.

The steady-state matrix $\boldsymbol{U}_\infty$ has the important property that

$$\boldsymbol{U}\boldsymbol{U}_\infty = \boldsymbol{U}_\infty\boldsymbol{U} = \boldsymbol{U}_\infty, \tag{3}$$

because steady-state means that the state variables no longer change. Suppose that not all columns in $\boldsymbol{U}_\infty$ are equal. Equation (3) implies that $\boldsymbol{U}_\infty$ is idempotent, or $\boldsymbol{U}_\infty^2 = \boldsymbol{U}_\infty$, and $\mathbf{1}$ is a left eigenvector of $\boldsymbol{U}_\infty$, so this requires that $\operatorname{tr}\boldsymbol{U}_\infty \geq 2$. However, the first m columns of $\boldsymbol{U}_\infty$ are identical and sum to one, so $\operatorname{tr}\boldsymbol{U}_\infty$ must be one. All rows of $\boldsymbol{U}_\infty$ must therefore be linearly dependent, and from the unit sum property, all columns must be identical. $\boldsymbol{x}_\infty = c\mathbf{1}$ is therefore the only possible steady-state solution, which contradicts our claim and concludes our proof.

□

Remark. Although the network converges under this condition, the source component alone, if there is one, determines the final value across the network. This is typically not desirable, as consensus suggests that all nodes should contribute to the final decision. In a real-world underwater acoustic sensor network, this is a plausible scenario, because sometimes, a subset of the network experiences severe difficulties decoding packets from others.

3.1 Quantisation effects

Mathematically, if we represent the opinion of node i as x_i, and the set of nodes, whose opinions node i has registered counting its own, as V_i, we find the compromise $\hat{x}_i$ as

$$\hat{x}_i = \frac{\sum_{j \in V_i} x_j}{|V_i|}.$$

The result is perturbed prior to quantisation to prevent stuck conditions in the network. Here, we explain how and why we perturb the state variables that make up the opinion.

The limited bits available in the JANUS ADB requires us to use discrete decision variables, which complicates the consensus problem. The work in [8] gives an example of a network that cannot reach consensus with discrete state variables without allowing nodes to swap values.

Swapping values requires reliable bidirectional links, which cannot be expected in underwater sensor networks. Instead, we may opt for a dithered quantisation scheme, in which a noise term ν, henceforth referred to as dither, is added to the updated state variable $x[t+1]$ before it is quantised and eventually transmitted [7]. The authors of [7] also state a sufficient condition on the dither for decoupling the quantisation error ϵ from the state variables $x[t]$: if the dither $\nu \sim \mathrm{U}(-\Delta/2, +\Delta/2)$ is iid for all time instances, where Δ is the resolution of the uniform quantisation scheme, then the quantisation error $\epsilon \sim \mathrm{U}(-\Delta/2, +\Delta/2)$ is also iid for all time instances. Here, $\mathrm{U}(a, b)$ denotes the uniform distribution between a inclusive and b exclusive. Thus, we perturb the compromise we find in the `adapt` state by adding dither to each state variable, choosing Δ to match the step size of each variable in the *channel estimate* message.

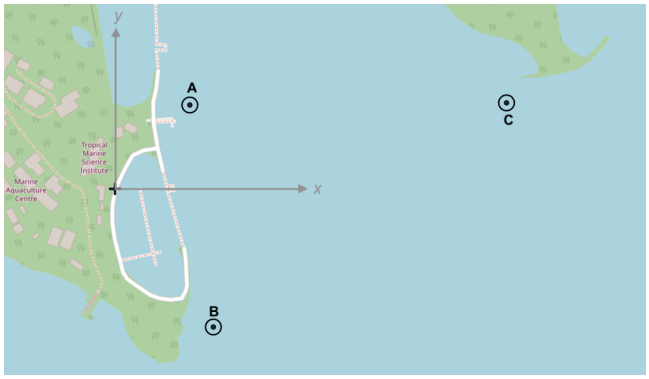

Figure 4: Map of the Netiquette testbed, reprinted with permission from [2, c. 5].

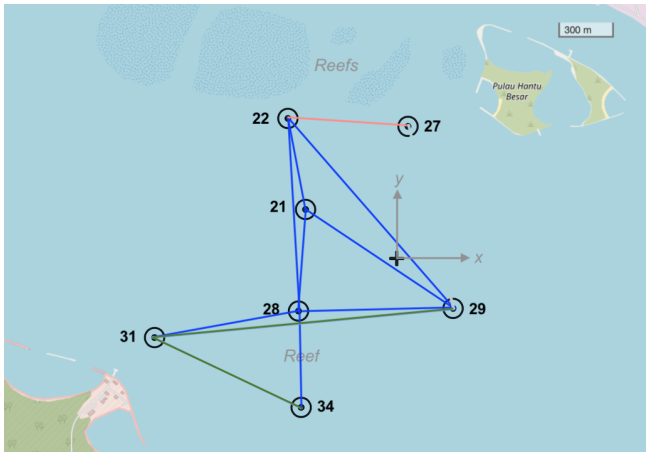

Figure 5: Map of the MISSION 2013 network, reprinted with permission from [2, c. 6].

4 IN SILICO EVALUATIONS

We implemented[2] the proposed protocol in UnetStack [2] and tested it by simulation, using a simulated version of the Netiquette testbed, a map of which is shown in Figure 4. The link A–B measures 371 m, A–C 530 m and B–C 616 m. It was also tested in a simulated version of the MISSION 2013 network, which is pictured in Figure 5. We also simulated the protocol in a bigger network, that can be representative of a realistic sensor network. To that end, we designed a 25-node network with a 5×5 grid topology. The nodes were spaced 1 km apart along each grid dimension. Each node decoded packets from its nearest horizontal and vertical neighbours with 90% probability of success, and always detected their packets. Nodes could also detect, but not decode, packets from its nearest diagonally adjacent neighbours. The simulated Netiquette testbed used the built-in default channel model. The MISSION 2013 model,

[2]The implementation is available at https://github.com/tuff-krister/janus-consensus

Table 2: Packet delivery ratios over all links in the MISSION 2013 model. Reproduced with permission from [2, c. 6].

From\To	21	22	27	28	29	31	34
21	—	0.926	0.266	0.917	0.912	0.000	0.552
22	0.867	—	0.471	0.751	0.850	0.000	0.288
27	0.359	0.381	—	0.313	0.322	0.000	0.000
28	0.847	0.869	0.390	—	0.845	0.925	0.863
29	0.539	0.693	0.333	0.688	—	0.374	0.000
31	0.000	0.000	0.000	0.902	0.805	—	0.795
34	0.236	0.436	0.000	0.684	0.000	0.544	—

which is a model of a seven-node network that was the core of an experiment in Singaporean waters, uses a simplified channel model. Table 2 shows the packet delivery ratio over each link.

For the sake of testing the here proposed protocol, we used $\beta = 1$ in the update step, as mentioned in Section 2.2. The packet loss ratio varied significantly over the course of the simulation, so the protocol was effectively simulated with both reliable and unreliable links.

Given that our focus is to check the consensus protocol that it converges rather than finds the holistically better OFDM parameters, each node generated a random opinion on the channel properties, distributed uniformly across the range of permitted values. The final opinions of all nodes were reported, as was the round number. The number of times each node broadcast an opinion, received an opinion, and failed to decode an opinion was also reported in the logs. Also the time from starting consensus to reaching consensus was reported. The logs were filtered and formatted to speed up data processing. Successful attempts were found by repeated row differencing, and these attempts could then be extracted from the processed logs. For the results in Section 4.1, we estimated the number of consensus attempts as the number of unique round numbers, and used it to find the number of unsuccessful attempts. For the results in Section 4.2, the reports from successful attempts were aggregated from all simulations of one selected time configuration per network. The reports from the first round of each simulation were discarded because the agents reported an incorrect time taken then.

4.1 Validation

This section shows how often the network reaches consensus for the different networks. To properly assess the functionality of our proposed protocol, we introduce the consensus error rate (CER) as

$$\text{CER} = \frac{\text{number of unsuccessful attempts}}{\text{number of consensus attempts}}, \tag{4}$$

which should be kept as low as possible.

We ran 20 simulations, each lasting for twelve hours in simulation time, and we repeated the simulations for various time configurations. By this, we mean the duration of the OFDM timer, which is the timer that each node sets on entering the done state, the minimum listening time t_0 and the rate parameter λ^{-1}. The CER was found by taking the weighted average of the CERs, setting the weights equal to the number of attempts per simulation.

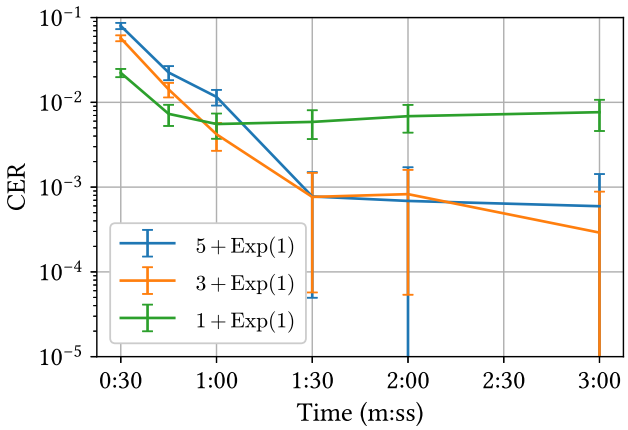

Figure 6: Consensus error rate plotted against the OFDM timer duration in the Netiquette testbed simulations.

Figure 6 shows the CER plotted against the OFDM timer duration in the Netiquette testbed (three nodes) simulations. Each series uses a different listening time, the duration of which is specified in the legend. Error bars mark 95% confidence intervals, assuming that the CER is Student's t-distributed with 19 degrees of freedom. The error bars of the $5 + \text{Exp}(1)$ series that extend below 10^{-5} extend all the way to zero, as does the error bar of the $3 + \text{Exp}(1)$ series at 180 s. We note here that the CER is higher when the OFDM timer is set to a lower duration in relation to the expected listening time. For example, the CER becomes as high as 0.080 ± 0.007 when the OFDM timer is set to 30 seconds and the listening time is distributed as $5 + \text{Exp}(1)$, whereas it falls to 0.022 ± 0.002 when the minimum listening time t_0 is set to 1 s. Setting the OFDM timer higher in relation to the waiting time causes the CER to decrease until it eventually reaches a "CER floor". The longer the expected waiting time is, the longer the OFDM timer can be before hitting this floor, and the lower this floor is located. The $1 + \text{Exp}(1)$ series has this floor near 0.007, and the other series have their error floor around an order of magnitude lower.

Figure 7 shows the CER plotted against the OFDM timer duration from the simulations of the MISSION 2013 network. The same principles with the legend and the error bars as in Figure 6 apply here. Worth noting here is that consensus is unlikely to be reached when the OFDM timer is too low, as the nodes do not have time to reach the done state before the next round begins.

Figure 8 shows the CER plotted against the OFDM timer duration from the simulations of the 5×5-grid network. Again, the same principles as the previous CER plots apply here. The error bars that extend below 10^{-4} extend all the way down to zero. This applies to the $15 + \text{Exp}(3^{-1})$ series at 25 minutes and the $10 + \text{Exp}(3^{-1})$ and $15 + \text{Exp}(5^{-1})$ series at 30 minutes. We notice that this network requires a very long OFDM timer duration to reach the CER floor. Two nodes in opposite corners of this network need to have a path of at least seven nodes to communicate to each other. Even so, there will be a vanishing fraction of the original opinion left in the final hop due to the adaptation step.

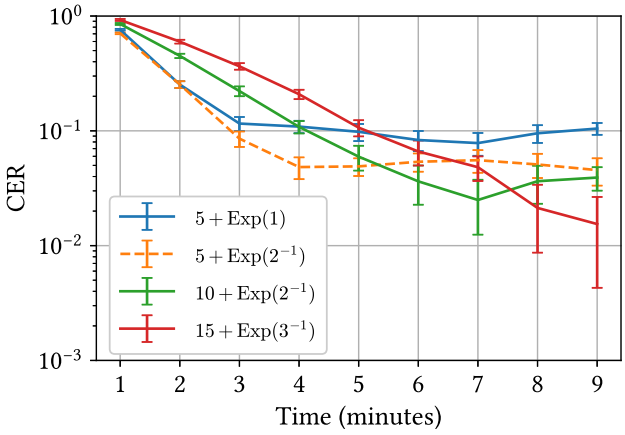

Figure 7: Consensus error rate plotted against the OFDM timer duration in the MISSION 2013 network simulations.

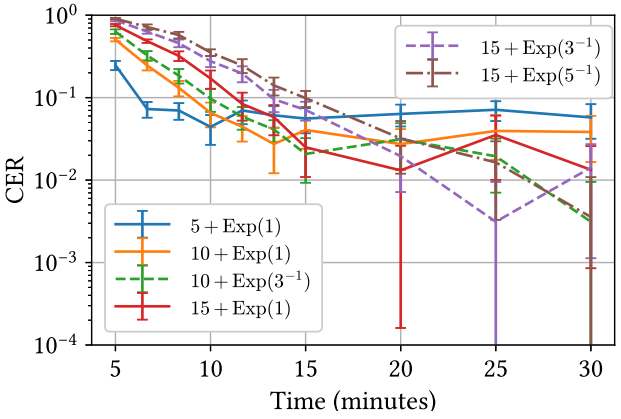

Figure 8: Consensus error rate plotted against the OFDM timer duration in the 5×5-grid network simulations.

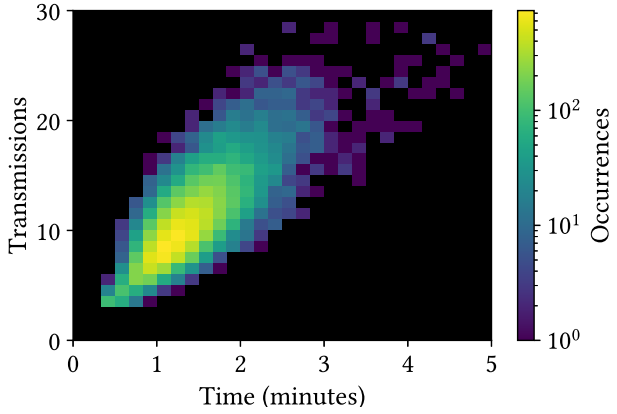

Figure 9: 2-D histogram of the time taken and JANUS packet transmissions made to reach consensus across the Netiquette three-node testbed.

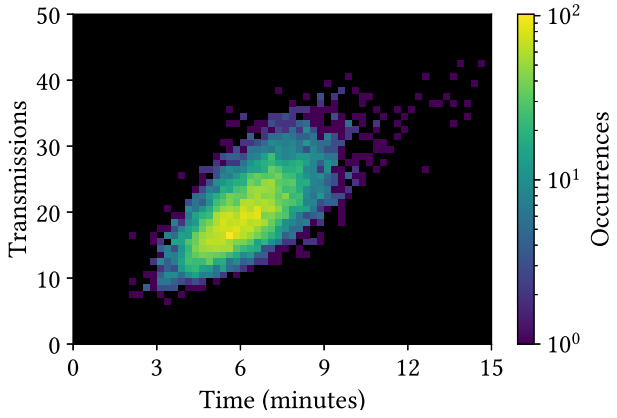

Figure 10: 2-D histogram of the time taken and JANUS packet transmissions made to reach consensus across the MISSION 2013 seven-node network.

4.2 Speed and energy

This section shows performance statistics in terms of the time taken to reach consensus and how many JANUS packets were sent over the course of each process. We re-used the simulation results that were reported in Section 4.1.

Figure 9 shows the joint distribution of the time taken per node and the number of JANUS packets transmitted per node of the 6 203 successful attempts in the three-node Netiquette testbed. We used the simulations where the OFDM timer was set to 1 minute, $t_0 = 5$ s, and $\lambda^{-1} = 1$ s, which Figure 6 shows to have a CER of 0.012 ± 0.002. There is a clear correlation between increasing time taken and more JANUS packets sent. This is expected, because the nodes wait after transmission to listen for peers' *channel estimate* packets. It occurred twice that a node reported that the attempt took longer than five minutes, or it transmitted more than 30 JANUS packets. The data from these reports are considered outliers and are not shown.

Figure 10 shows the joint distribution of the time taken to reach consensus and the number of JANUS packets transmitted for each node of the 1 227 successful attempts in the MISSION 2013 model. Here, we used the simulations where we let the OFDM timer be 5 minutes, $t_0 = 10$ s, and $\lambda^{-1} = 2$ s. There is again a correlation between time and transmissions, as expected. 30 attempts were considered outliers because they took longer than 15 minutes or transmitted more than 50 JANUS packets.

Figure 11 shows the joint distribution of the time taken to reach consensus and the number of JANUS packets transmitted for each node in the 5×5 grid, which successfully reached consensus in 332 cases. Here, we used the simulations where the OFDM timer was set to 20 minutes, $t_0 = 15$ s, and $\lambda^{-1} = 3$ s. Again, we see the expected correlation between transmissions and time taken. No attempt took longer than 40 minutes or transmitted more than 100 JANUS packets.

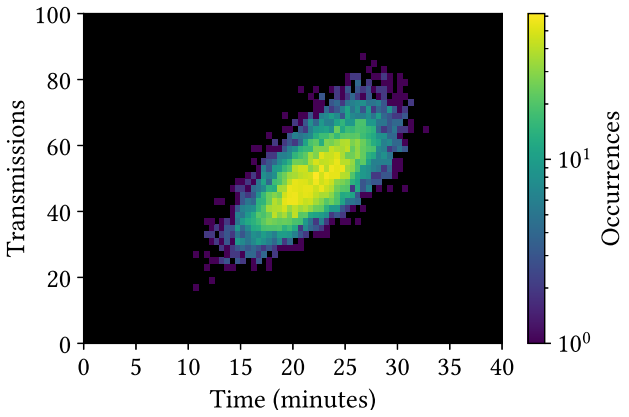

Figure 11: 2-D histogram of the time taken and JANUS packet transmissions made to reach consensus across the 5×5 grid.

4.3 Discussion

We made the best effort to ensure that all nodes could log their results at the end of each round, without cluttering the logs with spurious messages. This would happen if the node reported its results immediately on entering the done state. However, due to the asynchronous nature of an underwater sensor network, there were cases where not all nodes had reported their final values for some reason, such as missing a *new round* packet in the done state and then receiving a new opinion before the OFDM timer passed the halfway mark. An implication of the asynchronous nature of the network is thus that when the longest shortest path from one node to another consists of many hops, it is more likely that the more distant nodes are still not done when the OFDM timer expires on one node. As Section 4.1 shows, such cases can be encountered if the OFDM timer is set too low in relation to the expected listening time. They are counted as unsuccessful attempts for the purposes of this paper, and so contribute to CER as in (4). This is reflected by the decreasing trends in the CER graphs, which is seen rather clearly in Figure 7.

There were also attempts where the network did not reach consensus, but all nodes had time to report their final values. We attribute this to the asynchronous nature of the information exchange process, too, but primarily to packet loss. As Figure 6 indicates, the lower the expected time spent listening, the higher the CER floor becomes at a higher OFDM timer duration. This is a consequence of transmitting packets too frequently, because a node usually transmits a *channel estimate* message after it is done listening, and so it is more likely that packets collide.

5 CONCLUSION

We proposed a consensus strategy, built on JANUS, enabling underwater acoustic networks to reach consensus on which OFDM communication parameters should be used after the initial JANUS handshaking phase. We devise a necessary and sufficient condition for the network to converge without any constraints on the values of the consensus variables, and demonstrated in UnetStack that opportune implementations of this consensus protocol may work in a three-node network with a CER on the order of 10^{-3}, and a CER on the order of 10^{-2} in the MISSION 2013 network and the artificial 5×5-grid network.

As a next step, the protocol should be tested in real life conditions, so to assess how field conditions translate into deviations from the here proposed simulation results. From theoretical perspectives, moreover, we perceive the need for investigating which nonuniform discretisation strategies may better represent plausible values of the channel state variables, and aid thus achieving better cooperative choices on which OFDM parameters should be used.

ACKNOWLEDGMENTS

This project received funds from the Research Council of Norway, project number 302435, "Autonomous Underwater Fleets: from AUVs to AUFs through adaptive communication and cooperation schemes".

REFERENCES

[1] João Alves, Thomas Furfaro, Kevin LePage, Andrea Munafò, Konstantinos Pelekanakis, Roberto Petroccia, and Giovanni Zappa. 2016. Moving JANUS forward: A look into the future of underwater communications interoperability. In *OCEANS 2016 MTS/IEEE Monterey*. 1–6. https://doi.org/10.1109/OCEANS.2016.7761094

[2] Mandar Chitre. 2021. *Underwater Networks Handbook*. Retrieved 2022-10-23 from https://unetstack.net/handbook

[3] Morris H. DeGroot. 1974. Reaching a Consensus. *J. Amer. Statist. Assoc.* 69, 345 (1974), 118–121. http://www.jstor.org/stable/2285509

[4] Emrecan Demirors, George Sklivanitis, G. Enrico Santagati, Tommaso Melodia, and Stella N. Batalama. 2018. A High-Rate Software-Defined Underwater Acoustic Modem With Real-Time Adaptation Capabilities. *IEEE Access* 6 (2018), 18602–18615. https://doi.org/10.1109/ACCESS.2018.2815026

[5] Mari Carmen Domingo. 2012. An overview of the internet of underwater things. *Journal of Network and Computer Applications* 35, 6 (2012), 1879–1890. https://doi.org/10.1016/j.jnca.2012.07.012

[6] Fausto Ferreira, Roberto Petroccia, and João Alves. 2018. Increasing the operational safety of Autonomous Underwater Vehicles using the JANUS communication standard. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 1–6. https://doi.org/10.1109/AUV.2018.8729757

[7] Soummya Kar and José M. F. Moura. 2010. Distributed Consensus Algorithms in Sensor Networks: Quantized Data and Random Link Failures. *IEEE Transactions on Signal Processing* 58, 3 (2010), 1383–1400. https://doi.org/10.1109/TSP.2009.2036046

[8] Akshay Kashyap, Tamer Başar, and R. Srikant. 2007. Quantized consensus. *Automatica* 43, 7 (2007), 1192–1203. https://doi.org/10.1016/j.automatica.2007.01.002

[9] D.B. Kilfoyle and A.B. Baggeroer. 2000. The state of the art in underwater acoustic telemetry. *IEEE Journal of Oceanic Engineering* 25, 1 (2000), 4–27. https://doi.org/10.1109/48.820733

[10] L. Moreau. 2005. Stability of multiagent systems with time-dependent communication links. *IEEE Trans. Automat. Control* 50, 2 (2005), 169–182. https://doi.org/10.1109/TAC.2004.841888

[11] Roald Otnes. 2022. An underwater first contact method using JANUS. In *2022 Sixth Underwater Communications and Networking Conference (UComms)*. 1–5. https://doi.org/10.1109/UComms56954.2022.9905695

[12] Roberto Petroccia, João Alves, and Giovanni Zappa. 2017. JANUS-Based Services for Operationally Relevant Underwater Applications. *IEEE Journal of Oceanic Engineering* 42, 4 (2017), 994–1006. https://doi.org/10.1109/JOE.2017.2722018

[13] Roberto Petroccia, Gianni Cario, Marco Lupia, Vladimir Djapic, and Chiara Petrioli. 2015. First in-field experiments with a "bilingual" underwater acoustic modem supporting the JANUS standard. In *OCEANS 2015 - Genova*. 1–7. https://doi.org/10.1109/OCEANS-Genova.2015.7271740

[14] John Potter, João Alves, Dale Green, Giovanni Zappa, Ivor Nissen, and Kim McCoy. 2014. The JANUS underwater communications standard. In *2014 Underwater Communications and Networking (UComms)*. 1–4. https://doi.org/10.1109/UComms.2014.7017134

[15] Le Tian, Serena Santi, Amina Seferagić, Julong Lan, and Jeroen Famaey. 2021. Wi-Fi HaLow for the Internet of Things: An up-to-date survey on IEEE 802.11ah research. *Journal of Network and Computer Applications* 182 (2021), 103036. https://doi.org/10.1016/j.jnca.2021.103036

[16] Lin Xiao and Stephen Boyd. 2004. Fast linear iterations for distributed averaging. *Systems & Control Letters* 53, 1 (2004), 65–78. https://doi.org/10.1016/j.sysconle.2004.02.022