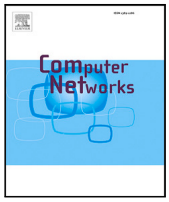
Computer Networks 244 (2024) 110345

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Experimental assessment of a JANUS-based consensus protocol

Emil Wengle [a,*], Elias Strandell Erstorp [b], Viktor Lidström [c], Damiano Varagnolo [d], Hefeng Dong [a]

[a] Norwegian University of Science and Technology (NTNU), Department of Electronic Systems, NO-7491, Trondheim, Norway
[b] Stockholm University (SU), Department of Geological Sciences, Stockholm, Sweden
[c] Swedish Defence Research Agency (FOI), Stockholm, Sweden
[d] Norwegian University of Science and Technology (NTNU), Department of Engineering Cybernetics, NO-7491, Trondheim, Norway

ARTICLE INFO

Dataset link: Askö Field Experiment Data 2023-05 (Original data)

Keywords:
JANUS
Underwater sensor network
Distributed algorithm

ABSTRACT

This paper proposes a distributed, JANUS-based protocol that enables an underwater acoustic network to reach consensus on arbitrary local opinions as numeric state variables. An envisioned scenario where nodes shall agree on parameters describing the acoustic environment is used to evaluate the protocol. The scenario exemplifies the protocol's potential in future applications where nodes use the environment description to decide on appropriate modulation and coding schemes.

The evaluation is based on numerical simulations and sea experiments in a challenging acoustic environment. The numerical simulations allowed examining the performance for different parameter values regarding the timing of transmission events and state transitions in the finite state machine implementation of the protocol. The best parameter configuration was used in the sea experiments conducted in a bay in the Baltic Sea. The experiments comprised several deployments of five to six commercial modems.

Results from the experiments show that the protocol can achieve a consensus up to 89% of the time in the tested environment, and up to 96% of the time if the state variables are permitted to differ by one discretisation step maximum across the network. In addition, if the network separates due to environmental conditions, connected components appear to achieve consensus more often when the links are more reliable. Finally, it is shown that when different consensus processes are active in parallel, packets from one process do not interfere with the opinions in different processes, besides the existing probability of packet loss due to packet collision.

1. Introduction

JANUS is an open standard for digital underwater acoustic communication developed by the NATO Centre for Maritime Research and Experimentation (CMRE) [1] with a wide range of applications in mind. In particular, it is intended to be a *lingua franca* to make first contacts and to discover the capabilities of other devices [2], to communicate distress signals, e.g., from submarines, or to enable switching the modulation and coding scheme, allowing the communication link to be adapted on the fly.

JANUS is inherently robust to the underwater acoustic channel, which is often defined by a harsh delay spread and Doppler spread, admitting communication in challenging sea conditions [1]. However, its data rate is limited to 36 bits/s[1] when using the 9.44 kHz to 13.6 kHz frequency band, which is insufficient in applications that involve streaming large amounts of data. JANUS may instead be used as a tool to find a sufficiently robust high-rate communication link, compatible with the deployed modems, to increase the network throughput. This approach was proposed in [2], where the authors designed a three-stage protocol: a node-discovery stage, a capabilities-sharing stage, and a negotiation stage (where a high-rate scheme can be selected). The three-stage protocol enables the network to select higher-rate links from a list, built on a peer-to-peer basis, based on the common capabilities that the peers found in the capabilities-sharing stage. Field experiments revealed the need for a more reliable link-switching mechanism since the modems could end up employing different communication links when feedback packets were lost. This behaviour is a known disadvantage with feedback-based link-switching mechanisms [3]. Despite this drawback, [2] clearly shows that changing the communication link dynamically requires solving two specific problems: *(1)* discovering the capabilities of the modems in contact and *(2)* performing the link-switching based on measured *feedback* on the current state of the channel between the modems. Using JANUS for both capability

* Corresponding author.
E-mail addresses: emil.wengle@ntnu.no (E. Wengle), elias.erstorp@geo.su.se (E. Strandell Erstorp), viklid@foi.se (V. Lidström), damiano.varagnolo@ntnu.no (D. Varagnolo), hefeng.dong@ntnu.no (H. Dong).

[1] The 36 bits/s data rate applies to baseline packets. For packets with cargo length approaching infinity, the data rate approaches 69 bits/s.

https://doi.org/10.1016/j.comnet.2024.110345
Received 20 September 2023; Received in revised form 28 February 2024; Accepted 16 March 2024
Available online 19 March 2024

discovery and link-switching requires a dedicated type of feedback messaging. While JANUS is the natural choice for link discovery, there being no other open standards, it may not be the best choice for such feedback and link-switching mechanisms, compared to other robust link solutions with higher data capacity. However, to the best of our knowledge, no other link-switching mechanism with equivalent technology readiness level to JANUS currently exists. Meanwhile, there are benefits to using JANUS for feedback and link-switching; its robustness is well-suited for retaining the topology after the discovery phase, which is key to efficient link-switching, and the JANUS packet definition allows flexibility in defining the feedback packets.

The goal of this paper is to assess the capabilities of JANUS as a generic protocol for performing distributed estimation. The paper extends the work of [4] by presenting a full network protocol that implements the capability-discovery step and enables the link-switching step described above via a distributed consensus algorithm. This is done by inserting feedback messages into certain JANUS packet fields and using the feedback to exchange state variables between the assets participating in the negotiation phase.

The proposed protocol is first evaluated in a simulated environment, then in field experiments, using a small network of five to six modems deployed in a shallow-water environment in the Baltic Sea, near Stockholm. The simulations are used to analyse the convergence rates of the protocol under typical operations, assuming sufficiently stable link performance. The field experiments are then used to assess how often the network achieves consensus in a real-world scenario, and, in the case of a disconnected network, how often the connected components can achieve a local consensus.

Related work. In [5], the authors use a subject-oriented business process management approach to underwater networking. One of their examples is a first-contact protocol, originally proposed in [6], where nodes exchange packets containing information about their own modulation and coding capabilities, as well as known neighbours. A node begins by transmitting a "hello, I am new" packet in JANUS. Other nodes that receive the "hello, I am new" packet and support a communication link that the sender also supports can reply to the sender using a common communication link. If the node shares no communication links with the sender, it replies with a "hello, I do not speak your language" packet in JANUS. The protocol can handle multiple co-existing communication links through translator nodes, which can reply with a "hello, I can help you" packet indicating which links it can translate. However, the packets only let nodes report two of 15 possible links (one of the values indicates "none").

The work in [7] devised a first-contact procedure using JANUS. New nodes send a first-contact request, detailing their known communication links (up to 15) and proposed address. The request is repeated periodically until a recipient delivers a response. The node then joins the network if the address is available and no further responses are heard in a set amount of time. If the address is occupied, and the following packet with occupied addresses reaches the new node, the new node selects an available address and restarts the procedure. The protocol allows nodes to resolve and assign unique 2-hop addresses and inform their peers how many, and which, links a new sender supports; however, the design of decision algorithms for negotiating the link-switching was left for future work since all modems supported the same links. Furthermore, response packets contain the CRC of the in-reply-to packet. If two nodes attempt to join the network simultaneously, there is a small chance that their first-contact packets generate identical CRCs. Hence, both nodes may mistake packets that were intended for the other node as intended for themselves. By letting new nodes select the initial 5-bit portion of the address at random, the risk of confusion is mitigated.

Consensus estimation has a wide range of applications to wireless sensor networks, including distributed target localisation [8], and clock synchronisation [9]. Consensus estimation can also be used in underwater wireless sensor networks, particularly for environmental sensing. One example is [10], wherein the authors consider a decentralised approach to estimating ocean currents. The network divides the ocean space into triangles and uses a distributed Kalman filter to track the ocean current field. Selected neighbours are considered in the update equation, which are included via a consensus matrix. The selection is based on shortest-path metrics from nodes of a specific type. The algorithm was tested with both reliable and unreliable links, and fixed node failures, albeit only in simulation. The simulated sensor network used acoustics for communication; however, the communication link was not reported, presumably because the algorithm was not run in any field experiment.

The work in [11] uses a criterion for consensus as motivation for a distributed algorithm that finds the connectivity degree of a directed graph, which the authors define as the least number of nodes that must be removed to break strong connectivity. The nodes start with knowledge of only their own existence, and add nodes and edges to their graph of known nodes and edges based on packets received from other nodes. These packets are broadcast in a fixed periodic pattern, which takes the propagation delay into account. The algorithm uses an auto-regressive moving-average method to track the reliability of the links, both when stationary and when varying over time. The algorithm is tested by simulation on a network of six nodes, with a specified probability of delivery per link. The paper shows that the network reaches a consensus on the network size after three cycles and the connectivity degree after ten cycles in the static case. However, a geographical topology of the simulated network is not provided, and the packet duration is not reported.

Organisation of the paper. Section 2 explains the proposed consensus protocol, in terms of behaviour and the packets it exchanges between participating nodes. Section 3 presents and motivates the numerical simulations that preceded the field experiments, and the decisions that were made based on the simulation results. Section 4 describes the experimental setup of the field experiments and the employed commercial modems. Section 5 explains how the data in the figures and tables were generated. Section 6 presents the field experiment results, in terms of validating the field experiment performance against the preparatory simulations, testing performance for varying transmission power, and running different consensus processes in parallel. Section 7 discusses and analyses the results from the field experiments. Section 8 concludes the paper.

2. Protocol structure and properties

This section describes the consensus protocol: how the exchanged packets are constructed by the sender and subsequently processed by the receivers. A mathematical analysis of its convergence properties is provided in Appendix.

2.1. Packet structure

Nodes employing the consensus protocol disseminate their opinion on a certain value by exchanging JANUS baseline packets with specific JANUS application data blocks (ADB, 34 bits) and application types. The data fields of the baseline packets are assigned as in Table 1 in [4], with one revision with respect to the application type. All packets have their JANUS *class user ID* set to 66 since this class user ID is not claimed [12]. Moreover, the packets from nodes that participate in a consensus process have their application type set to 0, to signal that the node must abort its consensus process via a *distress* packet, or to 1, to signal that the node is requesting to initiate a new round of consensus via a *new round* packet. A third application type is used to define specific consensus rounds in which only certain partitions of the nodes may wish to participate. In this case, they must choose the same application type number to avoid creating interfering opinions among the various partitions.

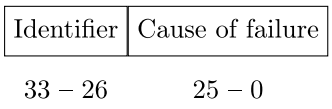


Fig. 1. Structure of a packet whose application type is 0, i.e., a *distress* packet.

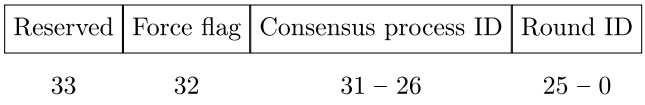


Fig. 2. Structure of a packet whose application type is 1, i.e., a *new round* packet.

Composition of the distress packet. The *distress* packet is sent only when a node cannot keep participating in the consensus process. This packet, shown in Fig. 1, contains the sender's identifier (8 bits), and uses the remaining bits to indicate what led the node to leave (e.g., the node running out of energy). If a node that is running a consensus process receives a *distress* packet, it drops any registered opinion received from the sender since the last adaptation step.

Composition of the new round packet. The *new round* packet is sent when a node requests to restart the consensus process. This packet is always sent when the done-timer expires. It may also be sent in application-specific circumstances; for example, in a link-tuning application, a *new round* packet may be sent when a node experiences too many packet drops, potentially because of a change in the properties of the communication channel. The proposed structure of a *new round* packet is shown in Fig. 2. Note the presence of the "forcing" flag, which allows a *new round* packet to interrupt a running consensus process. After being interrupted, the receiving node starts a new consensus process. If the forcing flag is not set, then the receiving node starts a new process only if in the `done` state. *New round* packets also accommodate a Consensus ID field (bits 31 down to 26) to indicate the targeted consensus process, so only nodes participating in a matching process may initiate a new round. Finally, the "round ID" field (last 26 bits) indicates which ID the new round has. The "round ID" number is initialised to 0 on the first round and is incremented every time the done-timer expires. If the node starts a new round due to receiving a *new round* packet, it sets its round ID to the "round ID" of the received packet instead.

Composition of the opinion packet. When the JANUS application type of the packet is 2 to 63, then the packet is an *opinion* packet, belonging to a consensus process ID specified by the application type. All *opinion* packets must contain the sender identifier (8 bits) encoded in bits 33 down to 26. The remaining 26 bits encode state variables, as determined by the consensus process ID. For the purposes of this paper, the state variables are encoded as following, for all valid application types (see also Fig. 3): delay spread (7 bits, $0, 1, \ldots, 127$ ms), Doppler spread (6 bits, $0, 0.5, \ldots, 31.5$ Hz), and noise level (7 bits, $32, 33, \ldots, 159$ dB re. 1 μPa). In both the simulations and the field experiments, the initial values of the three quantities were randomly selected from the supported values with equal probability of selecting each value, to better demonstrate the ability of the protocol to reach a consensus. For practical applications, the initial values of the quantities should be estimated from the most recent channel sounding, or from the most recent correctly decoded *new round* packet.

The packet structure in Fig. 3 provides the network with a basis for selecting an appropriate communication link, which is one of the envisioned applications of the protocol. The defined structure, shown as the coloured blocks in Fig. 3, may be tailored to other applications, subject to the following restrictions: *(a)* keep all state variables as numeric, *(b)* represent each variable as an n_i-bit wide integer or fixed-point number ($n_i \in \mathbb{N}$), signed or unsigned, *(c)* keep the mapping function of its representation as monotonically increasing, *(d)* keep all bits representing the same variable contiguous, i.e., occupying a closed interval in the ADB, *(e)* let any reserved bits be the most significant bits.

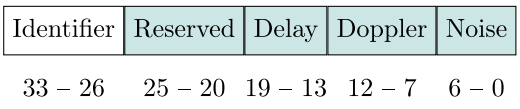


Fig. 3. Example structure of a packet whose application type is 2 to 63, i.e., signalling an *opinion* packet.

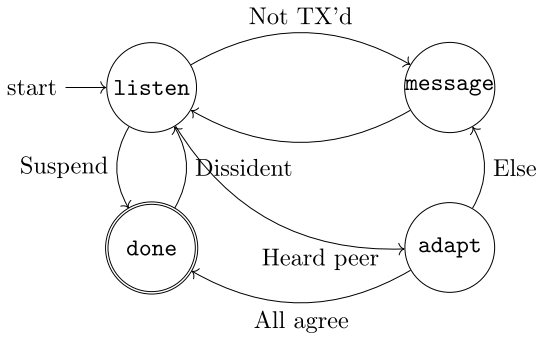


Fig. 4. State machine representation of the protocol.
Source: Adapted from [4].

2.2. Actions to be taken by the nodes when creating or receiving packets

This subsection exemplifies the protocol by providing a walk-through of the consensus process round, i.e., what actions are taken by the participating nodes in response to certain events. The walk-through here supersedes the walk-through in the prior work [4].

The protocol consists of four states: `listen`, `message`, `adapt`, and `done`. Each node goes through a sequence of cycles where it listens, broadcasts its current opinion, and adapts it when receiving information from others. A "pass" denotes one cycle of listening, broadcasting, and adapting. The protocol is visualised in Fig. 4, which shows a finite state machine representation of one consensus process round.

At the beginning of a consensus process round, all nodes begin in the `listen` state. Here, a node listens for incoming *opinion* packets for a random amount of time, sampled from a shifted exponential distribution with the probability density function ($u(t)$ being the unit step function)

$$p\left(t\,;\,t_0,\bar{t}\right) = \frac{u(t-t_0)}{\bar{t}} \exp\left(\frac{-(t-t_0)}{\bar{t}}\right). \tag{1}$$

The distribution in (1) is parameterised by an offset t_0 and the expected value of the exponential distribution $\bar{t}$. Any incoming *opinion* packets of the correct application type are registered on the node, overwriting any existing unprocessed opinion from the same sender. The listen-timer is reset, and its duration is sampled from (1) anew, every time the node registers a new opinion.

The first time in a round that the listen-timer expires, the protocol enters the `message` state. It may also enter this state if the extended listen-timer expires and the node still has not received any *opinion* packets of the correct kind in the current pass. Each time the protocol enters this state, the node transmits an *opinion* packet and then enters the `listen` state. When it does, the initial listen-time is extended to three times the generated time to account for other nodes resetting their listen-time.

A node can enter the `message` state three times in one pass (i.e., a cycle of listening, broadcasting and adapting). If this limit is reached in the first pass, and the node has not registered any *opinion* packets, then the node transmits its *opinion* as a last opportunity to reach out

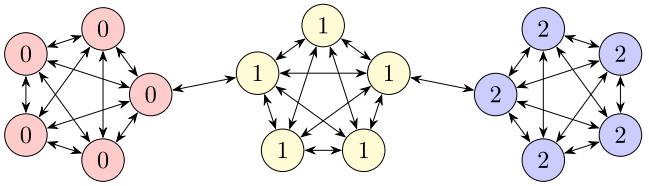


Fig. 5. Example of a deadlock situation when rounding to the nearest integer without dithering.

to any nodes that could be in reception range. After the transmission, the protocol goes to the `done` state, and finally suspends the process. This situation indicates that the node is likely isolated. If this situation occurs in a later pass, the protocol instead assumes that the network around it has reached a consensus already.

If a node has registered at least one opinion in the current pass when the listen-timer expires, it proceeds to the `adapt` state. First, it compares its opinion to all registered opinions. If all registered opinions are equal to its own, the node recognises that it has reached a local consensus. If at least one opinion is different to that of the node itself, it computes a compromise opinion from the registered opinions. The computation of such a compromise is user-definable, under the restriction that the compromise must lie on or within the convex hull of the set of known opinions in the opinion space. The implementation in this paper computes the compromise as the mean of the known opinions. The node then adjusts its opinion towards the compromise with a step length $0 < \beta \leq 1$, and rounds the new opinion towards the compromise. When the new opinion of the node is found, the protocol enters the `message` state, and the registry of state variables is cleared to ensure that the node only uses reasonably up-to-date opinions.

To decouple the rounding error from the state variables [13], and help the network escape deadlocks (for example, as shown in Fig. 5 for a modular network), the new opinion may be perturbed before rounding by dithering, i.e., adding noise to each variable, distributed uniformly between $-\Delta/2$ and $\Delta/2$, where Δ is the discretisation-step size of the variable. Dithering the opinion before rounding makes the quantisation scheme probabilistic, which lets the network reach a consensus almost surely [14].

To exemplify the mechanisms above, consider the network in Fig. 5, comprised by a linear network of three densely connected partitions, interconnected by only one bidirectional edge. Herein, the compromise opinion is computed as the mean of all registered opinions, and the number on each node indicates its initial opinion on an arbitrary integer state variable. Nodes with the same initial opinion are in the same partition, and have the same colour. Only a node that serves as a gateway to a node in a different partition (the rightmost 0-node in Fig. 5 is a gateway node because it is the only 0-node connected to a node in a different partition, namely the leftmost 1-node, and vice versa) can compute a compromise different to its own opinion. Each gateway node is connected to all other nodes in its own partition, and only one node in a different partition. Regardless of which packets a gateway node receives, the node will remain at their initial opinion after rounding the compromise. If half-integers are always rounded to the previous opinion, the network as a whole will never reach a global consensus. By dithering the compromise, there is a slight chance that gateway nodes change their opinions. This, in turn, implies a chance that other nodes in the same partition will change their opinions in the next pass, and the network may eventually reach a global consensus.

When a node recognises that it has reached a local consensus, the protocol broadcasts its opinion and enters the `done` state. Any actions defined by an active application type are taken at this point, and a done-timer with duration T is set. Meanwhile, the node continues listening for *opinion* packets. When registering an opinion of the same application type that is different from the local consensus, the protocol

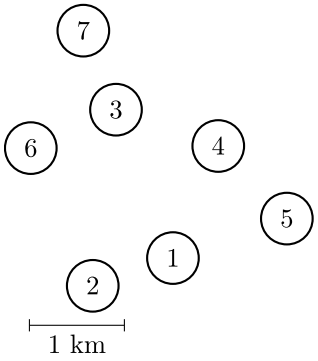


Fig. 6. Geographic distribution of the simulated network. Edges are omitted since the network was fully connected during the simulation.

initiates a new pass in fine-tuning mode. While in fine-tuning mode, the node forgoes any perturbation (dithering) in the adaptation step and rounds down the compromise when situated between two values. The motivation behind the fine-tuning mode is two-fold. Firstly, the other nodes in the network could have converged to a nearby value, and the node could either adjust its opinion down to their value or vice versa. Secondly, always rounding to the nearest permitted value can create deadlock situations; see Fig. 5.

When the timer reaches the halfway mark, the node logs the local consensus and is prohibited from engaging in new passes for the remainder of the current round. The time elapsed from initiating the round to the latest entry into the `done` state is also logged, as are packet transmission and reception statistics, plus the number of re-entries from the `done` state.

When the done-timer expires, or the node receives a *new round* packet with a higher round ID and the same consensus type in the `done` state, the node initiates a new round by broadcasting a *new round* packet, which is equivalent to any triggering *new round* packet.

Note that the user may specify additional events that may trigger a new round, specific to their elected application type.

3. Simulation setup

The protocol was simulated in the UnetStack framework [15] in advance of the field experiments. An important objective of the simulations was to find a suitable configuration of the timing parameters for a representative network topology; specifically, the minimum listen-time t_0, the mean of the exponentially distributed part of the listen-time $\bar{t}$, and the duration of the done-timer T, as defined in the explanation of the `done` state in Section 2.2.

For this reason, consider the following performance indexes:

- consensus error rate (CER), defined as the number of rounds of consensus that did not end in strict consensus, divided by the total number of rounds of consensus [4]; and
- quantised consensus error rate (q-CER), defined similarly, with respect to quantised consensus.

Strict consensus is attained when all nodes reach the same values of all variables, while quantised consensus permits two adjacent values of all variables across the network, as described in [16]. Hence, strict consensus also implies quantised consensus, so q-CER is necessarily less than or equal to CER. Whether quantised consensus is sufficient or not is application-dependent; for instance, a sensor network may accept quantised consensus on the oxygen concentration in a fish farm, but not always on which communication-link parameters to use.

Fig. 6 shows the geographic distribution of the simulated network. Edges are omitted since all nodes could receive packets from everybody else in the simulation. The high connectivity is a result of using

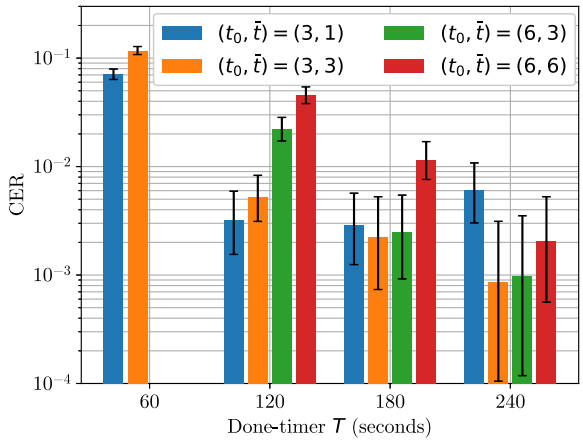


Fig. 7. Simulation consensus error rate plotted against the duration of the done-timer T.

UnetStack's default acoustic channel model, which does not depend on ray-tracing algorithms, but on a set of scalar channel parameters and a Rician/Rayleigh fading model, to determine the packet detection and decoding probabilities [15]. Such a configuration was then simulated 20 times, and each simulation ran the protocol for 12 h in simulator time. Note that the large number of simulations introduced a safeguard against random node failure, and introduced a natural partitioning of the simulation into cycles.

Fig. 7 shows the estimated (via simulation) CER as a function of the done-timer. The legend specifies the offset t_0 and shape parameter $\bar{t}$ of the distribution of the listening time, given in (1), and the error bars mark the 95% Clopper–Pearson intervals of the CER. The Clopper–Pearson interval, introduced in [17], is an exact confidence interval used in estimating the binomial proportion p of a binomially distributed random variable $X \sim \text{Bin}(n, p)$, when the sample size n is known. The Clopper–Pearson interval is exact in that it always has minimum 95% coverage, but it often has higher coverage [18], so terms like "with at least 95% confidence" are used herein.

From these results, one can note that setting too high listening-time parameters t_0 and $\bar{t}$ with respect to the done-timer T causes a large CER, because at least one node will often be out of the `done` state when the first done-timer on a node expires, and a *new round* packet arrives. Recall that a node may enter the `listen` state three times in one pass, listening for three times the normal duration each time. If no opinions are registered during any visit, then it will remain in the `listen` state for at least $9t_0$ seconds in total, and is expected to stay there for $9(t_0 + \bar{t})$ seconds. Simulating the $(t_0, \bar{t}) = (6, 3)$ and $(t_0, \bar{t}) = (6, 6)$ series with the 60-second done-timer was not motivated since the probability of listening for at least $T = 60$ seconds in a silent pass (i.e., a pass where a node registers no opinions) would become too high (99.845% and 99.979%, respectively). It is noted that increasing T while holding t_0 and $\bar{t}$ fixed reduces CER down to a floor. The location of the CER floor depends on two factors. First, the choice of t_0 and $\bar{t}$; the higher they are set, the lower the CER floor becomes. Second, because the modems access the channel randomly, extending the listen-timer reduces network load and mitigates packet loss due to collisions, lowering the floor. Missing repeated packets causes a node to suspend the process or recognise local consensus prematurely, often resulting in a consensus error if the node has time to report its local consensus.

In summary, these simulations suggested timing parameters $t_0 = \bar{t} = 3$ seconds and $T = 120$ seconds, which yielded the CER reported in Table 2. It should be noted, however, that the intermediate opinions were not logged in these simulations, preventing a transparent process. Consequently, the true CER may be lower; rounds where not all nodes had logged their final opinion were classified as erroneous, even if all nodes had reached the `done` state.

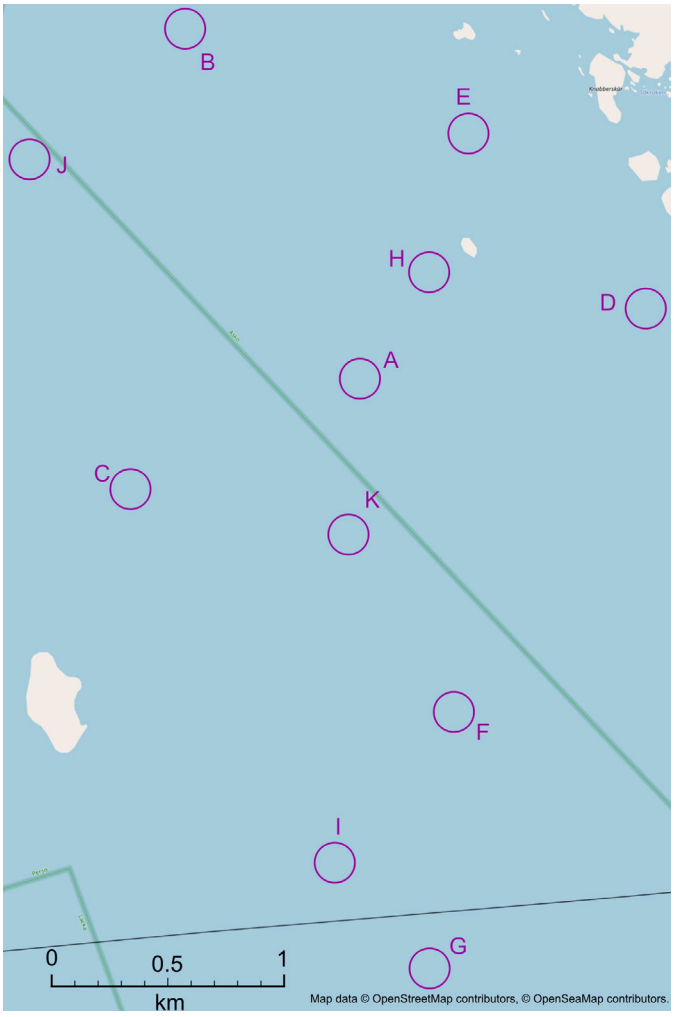


Fig. 8. Map over the experiment site, with node locations marked. Map data used under a CC-BY-SA 2.0 licence (link: https://creativecommons.org/licenses/by-sa/2.0/).

Table 1
Summary of the deployments.

Deployment	Key feature	Starting time (UTC)	Duration (h)
First	Verification	2023-05-08 13:00	4
Second	Verification	2023-05-09 11:00	24
Third	Varying power	2023-05-11 18:30	16
Fourth	Concurrent processes	2023-05-14 16:30	16

4. Field experiments

The field experiments were performed in a bay southwest of the Askö laboratory, located in the archipelago of the Swedish east coast. The water depth at the site varied between 10 m and 46 m. The network was deployed on four separate occasions, summarised in Table 1.

During the first two deployments, five Subnero Silver Edition modems were used to verify the simulation results from the prior work [4]. The modems were deployed at locations A through E, as marked in Fig. 8. During the last two deployments, a total of six Subnero Silver Edition modems were deployed. In the third deployment, the transmission power varied between cycles to emulate different input signal-to-noise ratios. The modems were deployed at locations A through D, plus F and G. In the fourth deployment, the transmission power was kept fixed and the network ran two concurrent consensus processes. Odd-numbered nodes ran one consensus process, and even-numbered nodes ran another consensus process, using a different

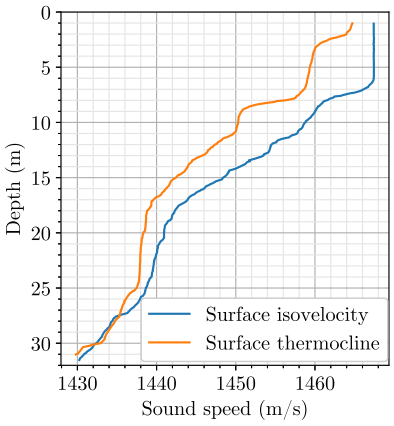


Fig. 9. Sound speed profiles measured two weeks after the field experiments, demonstrating the typical spring-time thermocline.

application type. The odd-numbered nodes were located at A, C and K, and the even-numbered nodes at B, H and J. All modems used a transmission power level of 175 dB re. 1 µPa @ 1 m, and the power level was scheduled to rotate between $0, -3, -6, -10, -14,$ and -20 dB relative to this power level at the start of each cycle in the third deployment.

The upper layer of water of the Baltic Sea is brackish, and the sound speed is primarily governed by the water's temperature, which varies drastically between seasons and the prevailing weather conditions [19]. The field experiments were performed in the spring, during which a thermocline generally forms in the upper layers of the water column due to solar radiation [20]. The weather was clear and calm during all field experiments, except for the last cycle of the final experiment when it turned cloudy. The wind came from the south and varied between 4 m/s and 6 m/s. For practical reasons, the sound speed profile was measured on-site two weeks after the field experiments; however, they demonstrate the typical behaviour of the regional spring-time thermocline. Fig. 9 shows two measurements of the sound speed profile. The profile has a negative gradient, so the acoustic waves refract towards the sea bottom, limiting horizontal long-range communication. The nodes were placed at depths between 2 and 3 m since a region of isovelocity appears in the upper layers due to water mixing whenever surface waves are present, as indicated in Fig. 9. The network operated overnight, under clear skies, during which the near-surface thermocline generally is weaker since the thermal heat flow driving the process is reversed. Consequently, the sound waves were likely less refracted at night, reflecting against the sea bottom fewer times and reducing the combined reflection loss.

During the field experiments, the modems used a UnetStack implementation[2] of the consensus agent from [4], revised to support additional consensus process IDs. In the field-tested version, the compromise function was defined as the average of all registered opinions, including the own opinion. A watchdog agent handled the experiment cycles and ensured that the consensus agents ran properly on the modems. Following the simulation results from Section 3, the listen-time parameters in (1) were set to $t_0 = \bar{t} = 3$ seconds and the done-timer to 120 seconds. These timing parameters were estimated to permit about 60 rounds per cycle and a consensus error rate (CER) less than 0.01.

[2] The implementation is available on the first author's GitHub (link: https://github.com/tuff-krister/janus-consensus), along with a generalised version of the startup script that was run on each device.

5. Data analysis

The consensus protocol writes the end-of-round results, as well as received and transmitted *opinion* packets, to a log file. The end-of-round result consists of the final value of the state variables; the number of transmitted, received, or lost packets; the timestamp of the report; the time and number of passes taken from start to finish; and the number of times it had reached a local consensus that was followed by a disagreeing *opinion* packet.

To facilitate the CER calculations, the end-of-round results from each modem's log file were aggregated and formatted, then sorted by timestamp. The modems were synchronised before deployment, and the clock drift was insignificant with respect to the duration of the experiments, so sorting the aggregated and formatted logs by timestamp did not mix results from different rounds of consensus. Successful rounds of consensus could then be filtered out by differencing the rows $n-1$ times, with n being the number of nodes used in the experiment, and locating the zeroes in state variables. The remaining rounds were classified by manual inspection. Reports from the same round were identified by comparing timestamps. Complete reports that were not classified as strict consensus were classified as a quantised consensus or no consensus by comparing the state variables before differencing, and incomplete reports could be classified as a strict consensus, a quantised consensus or no consensus by tracing the *opinion* packets in the log files near the relevant timestamps.

To produce the packet delivery ratio statistics, the log file on each modem was filtered to contain only reports of received and transmitted *opinion* packets, and windowed by timestamp. The *opinion*-packet transmissions and receptions were counted, separated by the sender ID. For each node pair (i, j), the packet delivery ratio from node i to node j was estimated as node j's reception counter of packets from node i, divided by node i's transmission counter. If $i = j$, the packet delivery ratio is one, because every node knows its own opinion.

The approach to produce the timestamp-received signal strength indicator (RSSI) diagrams is similar, but considers only notifications of received packets. UnetStack reports the data and RSSI of received packets, which enables estimates of the transmission loss over any link, provided the user knows the transmission power at the sending node. Only the reports of received *opinion* packets were considered, because only *opinion* and *distress* packets contain the sender ID, and no *distress* packets were sent. After filtering the *opinion* packet reports, the sender ID could be extracted from the application data block (ADB), allowing the user to create timestamp-RSSI scatter plots per incoming link.

6. Results

The protocol's performance was evaluated regarding the CER and associated Clopper–Pearson intervals and how these metrics are influenced by the modems' transmission powers (i.e., decreasing the SNR at the receivers). Specifically, the aim is to investigate the effects of transmission power on network topology, and in which subsets of the participating nodes that consensus is attained. Another question regards interference between two concurrent consensus processes, deploying two distinct networks with three modems in the same geographical area, where each network runs the consensus protocol with different application types.

6.1. Verification

The first two deployments were made to validate the consensus protocol. Table 2 shows the estimated CER and q-CER from the verification deployments, together with the simulations. The error rates are given in percent, with Clopper–Pearson intervals with at least 95% confidence in brackets. The number of completed consensus rounds, whether they ended with the network reaching consensus or not, is also reported. Note that the second deployment is presented both in its entirety, and

Table 2
Strict and quantised consensus error rate from the first two deployments and the simulations, with Clopper–Pearson intervals in brackets.

Deployment	Rounds	CER (%)	q-CER (%)
Simulation	3421	0.523 [0.312, 0.830]	0.497 [0.290, 0.794]
First	64	3.12 [0.381, 10.8]	1.56 [0.0396, 8.40]
Second (all cycles)	394	12.4 [9.34, 16.1]	2.28 [1.05, 4.29]
Second (node 4 active)	121	5.79 [2.36, 11.6]	3.31 [0.908, 8.25]
Second (node 4 muted)	273	15.4 [11.3, 20.2]	1.83 [0.597, 4.22]

Table 3
Strict and quantised consensus error rate of the third deployment, starting from the third cycle.

Component	Rounds	CER (%)	q-CER (%)
1236	54	54 [40, 67]	48 [34, 62]
123	54	17 [7.9, 29]	11 [4.2, 23]
136	92	52 [42, 63]	47 [36, 57]
45	90	23 [15, 33]	18 [11, 27]

partitioned by the functionality of node 4. At the beginning of the third cycle, node 4 ceased to transmit packets. Node 4 could receive packets from its peers, so it could still participate as a "listener".

In the first deployment, the network finished 71 rounds of consensus. In nine of them, results were not reported by all modems; in seven of these nine, the network was being recovered from the site. These seven rounds are not counted towards the total in Table 2. In one of the remaining two rounds, one node had locked its opinion to local consensus because it failed to decode most of the packets from its peers. One of its variables differed by one increment from the rest of the network, and the network only reached quantised consensus as a result. Also, the node that could not report its local consensus was still in the time window where it could initiate another pass in fine-tuning mode. While in this window, it missed all *new round* packets before it heard its first *opinion* packet, triggering the "Dissident" transition in the state machine representation in Fig. 4. In the other of those two rounds, one node recognised local consensus prematurely, and it lost several packets from its peers before it reported its results. Consequently, this node requested a new round before the rest of the network had finished, causing a consensus error.

In the second deployment, the network conducted six 4-hour cycles and 400 rounds of consensus in total. Six of these rounds ended as the network was recovered, and are not counted. The network as a whole successfully completed 345 rounds of consensus, of which four demanded closer inspection because not all nodes had reported the end-of-round results. This gives 49 rounds that ended in a strict consensus error. Node 4 was one increment away from consensus in 34 of the rounds, indicating quantised consensus. The 15 remaining rounds ended in a consensus error due to multiple packets lost at inappropriate times, except in one case, where one node generated a very long listening-timer when the rest of the network had reached consensus. Six of these 15 rounds ended with at most two adjacent values in all state variables, which qualifies for quantised consensus, but not strict consensus.

6.2. Varying signal power

Fig. 10 shows cycle-wise estimates of the packet delivery ratios over all links in the network. It is seen in Fig. 10(a) that the network was separated in the first cycle, as nodes 4 and 5 were isolated from the rest. The overall connectivity was better in the second cycle (Fig. 10(b)). Node 5 was isolated from all nodes but node 4, which served as a gateway to the rest of the network. The network topology was similar in the first and third cycles (Fig. 10(c)); the main differences are in the link from node 3 to node 1, and the links between node 2 and node 6. Node 3 rebooted during the third cycle, and reset its power cycle. The increased source level let node 1 receive more packets from node 3, which raised the packet delivery ratio over that link, and made that link asymmetric. In the fourth cycle (Fig. 10(d)), node 4 and node 5 were completely separated from the rest, and the connected component between the other nodes became centralised, with node 1 at the centre. Also, node 2 lost connection to the rest of its component halfway into the fourth cycle, and was effectively removed from the network.

The global consensus error rate was very high in all cycles but the second cycle, which is a consequence of the poor network connectivity in these cycles and the occasional absence of a node. On the other hand, the connected components, shown in Fig. 11, could reach consensus among themselves. These components will be the focus of the study, rather than the whole network.

Table 3 shows the CER, the q-CER, and the number of rounds that were completed in the third and fourth cycles, organised by the connected components. Consecutive rounds where a node in the component did not attempt to participate are not counted.

6.3. Concurrent consensus processes

The network ran one consensus process on nodes 1, 3 and 5, and the other consensus process, with a different consensus process ID, on nodes 2, 4 and 6.

Fig. 12 shows the packet delivery ratio over all links between functional nodes in the network. Nodes 4 and 5 failed to participate in the process due to corrupted files, and are omitted. Node 1 could not detect any packets across any link, and always suspended its process in the first round of each cycle. However, the rest of the network could receive its *opinion* packets. Node 3 heard node 1 in only one consensus process round, and finished that round normally. Naturally, the round ended in a consensus error, both strict and quantised. All *opinion* and *new round* packets that node 3 received from nodes 2 and 6 were correctly ignored because they were of the wrong consensus process ID, and vice versa. Nodes 2 and 6 finished 87 rounds of consensus, at a CER of 8.05% [3.30%, 15.9%], and a q-CER of 4.60% [1.27%, 11.4%].

7. Discussion

This section analyses and discusses the outcome of the field experiments, and aims to answer the questions posed in Section 6.

7.1. Verification

The first two deployments served to verify the consensus protocol and assess its performance with respect to CER and q-CER. From Table 2, both the CER interval ([0.381%, 10.8%]) and the q-CER interval ([0.0396%, 8.40%]) from the first deployment overlap with the Clopper–Pearson intervals from the simulation ([0.312%, 0.830%] and [0.290%, 0.794%], respectively). This indicates that there was no statistically significant difference in performance between the simulation and the verification deployment, even though two additional nodes were used in the simulation. It should be noted that the first deployment lasted for one cycle of four hours, and admitted many fewer rounds of consensus than the simulations with the same timing-parameter configuration (240 h effective time). Accordingly, the Clopper–Pearson interval for such a cycle is much wider.

The second deployment lasted for six cycles of four hours each. From Table 2, neither the CER nor the q-CER interval across all cycles from the second deployment overlaps with the corresponding intervals from the simulation, [9.34%, 16.1%] and [1.05%, 4.29%] vs. [0.312%, 0.830%] and [0.290%, 0.794%], so there is a statistically significant difference in performance between these two deployments. The three main potential contributing factors to this statistically significant difference appear to be:

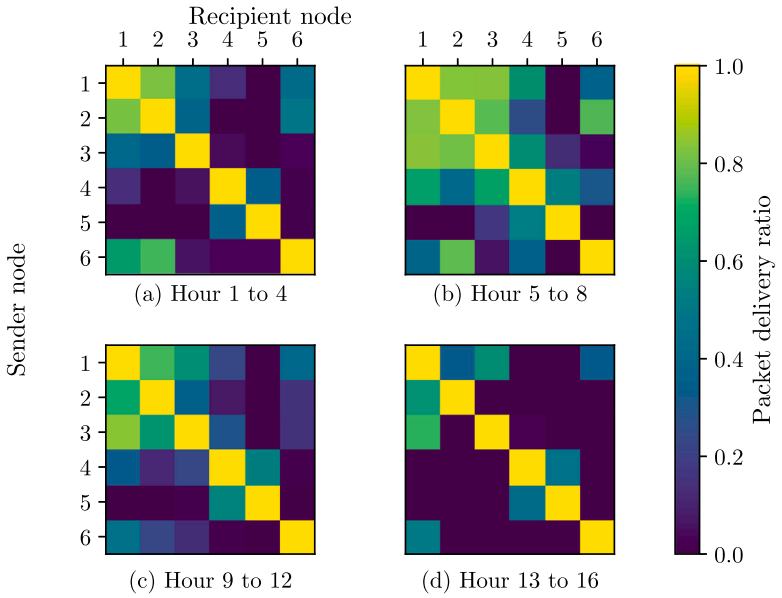


Fig. 10. Packet delivery ratio in the network in the third deployment, per cycle.

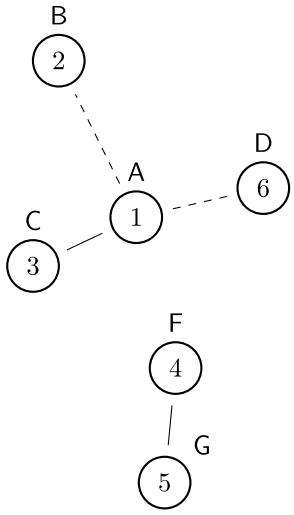


Fig. 11. Topology of the connected components from the third cycle and onward. The letters correspond to locations in Fig. 8 and the dashed lines indicate weak links.

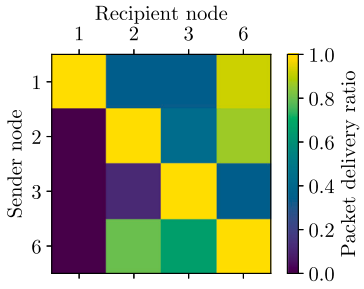


Fig. 12. Packet delivery ratio in the functional part of the network in the fourth deployment.

1. in general, the underwater acoustic channel is time-varying, and even when the weather conditions are similar on two days, the channels on those days could be quite different [21];
2. underwater channel simulators cannot be perfect statistical representations of real world channels, and there is no consensus on how to model the underwater channel statistically — see [22] and references therein;
3. node 4 stopped transmitting packets at the beginning of the third cycle, which reduced the probability of reaching strict consensus.

To support the considerations above, consider the bottom two rows of Table 2. In the first two cycles, when node 4 could transmit packets, the CER was 5.79% ($[2.36\%, 11.6\%]$), which still is outside the Clopper–Pearson interval from the simulations. In the other cycles, when node 4 could not transmit, the CER was 15.4% ($[11.3\%, 20.2\%]$), which barely overlaps with the Clopper–Pearson interval from the first two cycles. Due to the marginal overlap, it cannot be said with at least 95% confidence that the transmission failure alone had a significant impact on CER. The corresponding intervals for q-CER before and after the failure of node 4 overlap more clearly ($[0.908\%, 8.25\%]$ vs. $[0.597\%, 4.22\%]$). The network did not become separated when the outgoing links from node 4 were removed, which suggests that the protocol could be tolerant to node failures, provided the failure does not disrupt the rooted structure of the network, and quantised consensus is sufficient for the application at hand.

7.2. Varying signal power

The packet delivery ratio across most links was the best in the second cycle (Fig. 10(b)), surpassing even the performance in the first cycle (Fig. 10(a)). The improved connectivity may seem counter-intuitive, because the transmission power was 3 dB lower in the second cycle, compared to the first cycle. Reducing transmission power should also reduce SNR, and reducing SNR typically reduces packet delivery ratio. Recall, however, that the second cycle took place at night, so the thermocline was possibly less adverse than in the first cycle. The less adverse thermocline could be reflected in the received signal strength indicator (RSSI) of successfully received packets.

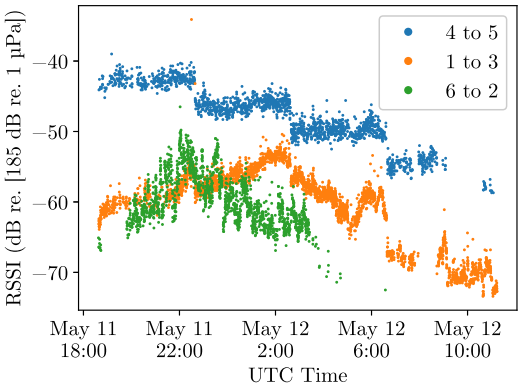


Fig. 13. Estimated RSSI of successfully received packets over select links.

Fig. 13 shows the RSSI of successfully received packets, as estimated by the physical agent on UnetStack, of select links. Inspecting the RSSI of the received packets reveals that the transmission loss over many links varies more quickly, and by a larger magnitude, than the transmission power. The link from node 4 to node 5 was one of the stablest in terms of transmission loss, and the sudden decrements were caused by cycling the transmission power. In contrast, the link from node 1 to node 3 clearly varies over time in transmission loss. The transmission loss decreases until 02:00 UTC, and then increases to the end of the deployment, except for a brief period after 05:00 UTC, where it decreases. The sun set at 19:04 UTC and rose at 02:27 UTC; these times correspond rather well with the UTC time of the observed trend changes of the $1 \rightarrow 3$ link. As was argued in Section 4, solar radiation tends to form a surface thermocline. As the sun set, the upper layers of the water column might have cooled, weakening the surface thermocline, and allowing the packets to travel farther horizontally without reflecting. Conversely, as the sun rose, the upper layers might have heated again, forming a surface thermocline. The time-varying transmission loss has a greater impact on the link from node 6 to node 2. The transmission loss over the $6 \rightarrow 2$ link varies more quickly than the loss over the other links on a short time scale, and the trend is decreasing until 22:00 UTC, then increasing, to the point that the link is effectively broken near 03:00 UTC.

The time-varying links affected the network topology, and from the third cycle and on, the network consisted of two connected components. Fig. 11 shows that nodes 4 and 5 formed one component, and the others formed the other component.

Table 3 indicates that there is a statistically significant difference in the performance of two components, with at least 95% confidence. From Figs. 10(c) and 10(d), which show the packet delivery ratio over these two cycles, in order, the links between node 6 and the rest of its component were poor or broken, in both directions. Indeed, node 6 rarely decoded more than five packets in one round, and node 6 suspended its round or had a different opinion in all but one of the attempts that resulted in a quantised consensus error. Removing node 6 from the 1236 component, the consensus error rates decrease significantly, as Table 3 shows. The improvement in performance with respect to the entire component (123 vs. 1236) is statistically significant with at least 95% confidence, suggesting that the poor link to node 6 is the culprit. The same conclusion can be made if node 2 is ignored in its component, because node 2 always reached at least a quantised consensus with node 1 from the third cycle and on. In contrast, node 4 and node 5 achieve a packet delivery ratio near 0.5 between each other, and, as the series "4 to 5" in Fig. 13 shows, these links have a stable transmission loss (the reversed link, from node 5 to node 4, is very similar).

In the fourth cycle, node 2 could receive packets exclusively from node 1. The link eventually ceased to provide successfully decoded packets, and node 2 was fully isolated. It was also observed that the connected components of the network experienced periods of outage. As no packets were successfully delivered, the outage eventually caused the connected components to suspend their processes, and they remained suspended for the remainder of the cycle. A memory mechanism would be a more effective way of handling outage situations. If a node has successfully received a packet at any point in a previous round, then ends a round without having received any packets, it still suspends the process, but it initiates a new round after a set amount of time has elapsed. The time to wait for next round should be longer than the done-timer, preferably by a factor near ten.

7.3. Concurrent consensus processes

One node in each partition failed to participate due to missing files, and the partition consisting of odd-numbered nodes immediately suspended their consensus rounds because node 1 did not detect any packets. Still, node 3 could receive four *opinion* packets from node 1 in one round, and many packets from nodes 2 and 6. Node 3 correctly ignored all *opinion* and *new round* packets that were delivered from the even-numbered nodes, so the opinion of node 3 was unaffected by them in the round that it could finish, even though it finished with a quantised consensus error. Likewise, the even-numbered nodes ignored all *opinion* packets from the odd-numbered nodes. The logs therefore showed that consensus processes with different consensus process IDs can be run in parallel in a network, without interfering with the opinions of each other.

8. Conclusion

This paper presented and explained the working mechanisms of a consensus protocol built on JANUS that enables nodes in an underwater network to decide which modulation and coding scheme they should use after having used the protocol as a first contact. The protocol's performance was assessed both in numerical simulation and in field experiments conducted in a bay southwest of the Askö laboratory, Sweden, using five to six Subnero Silver Edition modems. The field experiments confirmed that the protocol can operate on Subnero modems, and gives acceptable performance even in a challenging acoustic environment. However, the challenging environment caused a noticeable discrepancy in performance between simulation and field tests. The tests further indicated that, due to the time-varying transmission loss and transmission power, network partitions which have more reliable connectivity are more likely to reach a local consensus, and that two coexisting networks using different consensus IDs may operate in parallel without interfering with each other, despite poor connectivity and node failures. Overall, simulations and field tests indicate the suitability of the protocol for the initially intended uses.

It is envisioned that the protocol may be extended to support JANUS' link-switching application by using actual measurements of the channel, rather than assigning them random numbers. In other words, extending the protocol to use the consensus on these measurements to select a predetermined parametric communication link, such as orthogonal frequency-division multiplexing (OFDM), with parameters chosen in a distributed-optimisation sense to give the "best" data rate with a sufficiently high reliability.

Another potential future extension is to expand the possibility of running multiple consensus processes in parallel, to acquire a distributed network-partitioning algorithm. For instance, based on packet delivery ratio estimates, the protocol can be extended to find network partitions that are more strongly connected (and possibly overlapping). The protocol may subsequently initiate one consensus process per partition.

CRediT authorship contribution statement

Emil Wengle: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Elias Strandell Erstorp:** Methodology, Resources, Software, Validation, Writing – original draft, Writing – review & editing, Investigation. **Viktor Lidström:** Methodology, Resources, Software, Writing – original draft, Writing – review & editing, Investigation. **Damiano Varagnolo:** Conceptualization, Formal analysis, Funding acquisition, Project administration, Supervision, Visualization, Writing – original draft, Writing – review & editing. **Hefeng Dong:** Funding acquisition, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is available on Mendeley Data, under the name Askö Field Experiment Data 2023-05.

Askö Field Experiment Data 2023-05 (Original data) (Mendeley Data).

Acknowledgments

This work was supported by the Research Council of Norway, grant number 302435: "Autonomous Underwater Fleets: from AUVs to AUFs through adaptive communication and cooperation schemes", and by the Swedish Foundation for Strategic Research (SSF) via grant number IRC15-0046: the Swedish Maritime Robotics Centre (SMaRC). The authors would also like to thank Prof. John Potter for his valuable feedback on the experiment design and the manuscript.

Appendix. Convergence analysis

The consensus protocol is distributed in nature; meanwhile, there is a nonzero probability of losing packets over the links in a typical underwater sensor network. Hence, the relevant question is whether an underwater sensor network running the consensus protocol achieves probabilistic consensus, as defined in [23]. This section is devoted to analysing the convergence of such a network, where the channel is assumed to be stable for a long duration with respect to the time it takes to reach a consensus.

Let $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k])$ be the graph representation of the network at time instant k. The vertex set $\mathcal{V}$ is deterministic and has cardinality n. The edge set $\mathcal{E}[k]$ is random, such that the edge $(i,j) \in \mathcal{E}[k]$ with probability p_{ij}, where we define $p_{ii} = 0 \,\forall\, i$. The probabilistic framework captures both the lossy links and the finite listening time per node, i.e., packets may either be lost at the receiver, or arrive at the receiver too late to be included in any given pass.

The adjacency matrix $\boldsymbol{A}[k]$, where $a_{ij} = 1 \iff (i,j) \in \mathcal{E}[k]$, can thus be modelled as an iid sequence of $n \times n$ Bernoulli matrices with probabilities $\boldsymbol{P}$. To facilitate the analysis, we introduce the communication matrix $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{I}$, which is equivalent to adding a self-loop at each vertex in $\mathcal{G}$. We refer to row i of matrix $\boldsymbol{A}$ as $\boldsymbol{A}_{i*}$, and to column j of $\boldsymbol{A}$ as $\boldsymbol{A}_{*j}$.

Let $\boldsymbol{x}[k] \in \mathbb{R}^n$ be a column vector that contains the state variables of all nodes, and q be the time shift operator, such that $q\boldsymbol{x}[k] = \boldsymbol{x}[k+1]$. Let also $\boldsymbol{u}[k] \in \mathbb{R}^n$ be a column vector of iid activation variables, where $u_i[k] \sim \mathrm{Be}(\upsilon) \,\forall\, i$, whose role is to signify which elements of $\boldsymbol{x}$ are to be updated. We will omit the time index k when doing so does not raise confusion.

We make the following assumptions.

1. When all edges with $p_{ij} > 0$ in $\mathcal{E}$ exist, $\mathcal{G}$ is rooted, *i.e.,* there exists an $r \in \mathcal{V}$ for which the oriented path (r, υ) exists for all $\upsilon \in \mathcal{V} \cap \{r\}^C$. By Theorem 3 in [24], this is a necessary and sufficient condition for reaching consensus in finite time.
2. Also when all edges with $p_{ij} > 0$ exist, $\mathcal{G}$ is a directed acyclic graph, or can be reduced to one if it is not. By [23, Corollary 3.2], a strongly connected component with self-loops can reach consensus, because all vertices in a strongly connected component have oriented paths to each other.
3. The update step computes the exact average of the state variables that were successfully received. This is the update policy that we use in the field experiments, without quantisation and perturbation.

Under these assumptions, the update equation for vertex i is

$$qx_i = \begin{cases} \frac{(\boldsymbol{C}_{*i})^{\mathrm{T}}\boldsymbol{x}}{(\boldsymbol{C}_{*i})^{\mathrm{T}}\boldsymbol{1}} & \text{w. p. } \upsilon, \\ x_i & \text{w. p. } 1-\upsilon; \end{cases}$$

across all of $\mathcal{G}$, the update equation becomes

$$q\boldsymbol{x} = \boldsymbol{F}[k]\boldsymbol{x} \tag{A.1}$$

$$= \left((\boldsymbol{I} - \operatorname{diag}\boldsymbol{u}) + (\operatorname{diag}\boldsymbol{u})(\operatorname{diag}(\boldsymbol{C}[k]^{\mathrm{T}}\boldsymbol{1}))^{-1}\boldsymbol{C}[k]^{\mathrm{T}}\right)\boldsymbol{x}, \tag{A.2}$$

where $\boldsymbol{F}[k]$ is a sequence of stochastic matrices.

From Assumption 2, we may permute the vertices in $\mathcal{G}$ such that $\boldsymbol{C}$ becomes triangular. An important property of triangular matrices is that their eigenvalues are given by the diagonal elements. We can, therefore, impose an upper bound on the expected rate of convergence by studying the expected diagonal of $\boldsymbol{F}$, and considering the second largest expected eigenvalue (the largest one having to be one if the process describes a consensus one [24]).

Taking the expected value of $\operatorname{diag}\boldsymbol{F}$, where $\operatorname{diag}\boldsymbol{C}$ is a column vector consisting of the diagonal elements of the communication matrix $\boldsymbol{C}$, results in

$$\mathrm{E}\,(\operatorname{diag}\boldsymbol{F}) = (1-\upsilon)\boldsymbol{1} + \upsilon\,\mathrm{E}\Big(\operatorname{diag}\left(\boldsymbol{C}[k]^{\mathrm{T}}\boldsymbol{1}\right)^{-1}\operatorname{diag}(\boldsymbol{C}[k])\Big). \tag{A.3}$$

Because $c_{ii}[k] = 1 \,\forall\, i,k$, the factor $\operatorname{diag}\boldsymbol{C}[k] = \boldsymbol{1}$ is deterministic, and can be moved out of the expectation operator. To determine the expected value (A.3), it remains to determine $\mathrm{E}\left(\operatorname{diag}(\boldsymbol{C}[k]^{\mathrm{T}}\boldsymbol{1})^{-1}\right)\boldsymbol{1}$. The factor $\boldsymbol{c} = \boldsymbol{C}[k]^{\mathrm{T}}\boldsymbol{1}$ is a column vector of sums of independent and not identically distributed Bernoulli variables. Each of its elements can therefore be expressed as a Poisson binomial variable, where the terms of element m have probabilities

$$p_{jm} \in \begin{cases} [0,1], & j < m; \\ \{1\}, & j = m; \\ \{0\}, & j > m. \end{cases}$$

Note that the expression for the probabilities associated with the cases for which $j \geq m$ follows from Assumption 2 and that each vertex always knows its own x_m. This gives element c_m a total of $m-1$ effective terms and one constant unit term, so that we can compute its probability mass function as [25]

$$\Pr(c_m = \mu) = \begin{cases} \displaystyle\sum_{\mathcal{X} \in \mathcal{F}_{\mu-1}^{m}} \left(\prod_{i \in \mathcal{X}} p_{im} \prod_{j \in \mathcal{X}^C} (1 - p_{jm}) \right) & \text{if } 1 \leq \mu \leq m; \\ 0 & \text{otherwise,} \end{cases} \tag{A.4}$$

where $\mathcal{F}_k^m$ is the set of $\binom{m-1}{k}$ possible choices of k integers satisfying the condition $1 \leq n \leq m-1$. Because the expressions above involve the inverse of $\operatorname{diag}\boldsymbol{c}$, we must find $\mathrm{E}(c_m^{-1})$ for all m. Using (A.4) and the

definition of the expected value of a discrete random variable, we find that

$$\mathrm{E}(c_m^{-1}) = \sum_{\mu=1}^{m} \mu^{-1} \Pr(c_m = \mu). \tag{A.5}$$

Letting $m = 1$ in (A.5) gives the expectation of the first diagonal element, which is unity by definition. For the consequent possible values of m, we note that each $\mathrm{E}(c_m^{-1})$ is upper bounded by

$$\mathrm{E}(c_m^{-1}) \leq \Pr(c_m = 1) + \frac{1}{2}\Big(1 - \Pr(c_m = 1)\Big), \tag{A.6}$$

which follows from taking the expected value of a decreasing function of c_m. We note that this bound is tight for $m = 2$, and tighter bounds can be achieved by including more $\Pr(c_m = \mu)$ terms in (A.6). Furthermore, from Assumption 1, there is necessarily at least one $p_{jm} > 0, j < m$ for each $m > 1$; hence, the upper bound on (A.5) is necessarily smaller than one for those m's. If there were no $p_{jm} > 0$ for a given m, then $\mathcal{G}$ would not be rooted, the unit eigenvalue of $\boldsymbol{F}[k]$ would not be unique, and probabilistic consensus would not be guaranteed.

Using the bound in (A.6), and the fact that the second largest eigenvalue of $\boldsymbol{F}$ determines the convergence rate, the convergence rate of (A.1) is bounded by

$$\lambda_{\mathrm{M}} = \max_{m>1} \mathrm{E}(\operatorname{diag} \boldsymbol{F})_m \leq 1 - \upsilon + \upsilon \max_{m>1} \frac{1 + \prod_{\mu=1}^{m-1}(1 - p_{jm})}{2}. \tag{A.7}$$

It shall be remarked that determining the convergence rate is more challenging when strongly connected components are present in $\mathcal{G}$, whereby the update matrix $\boldsymbol{F}$ will no longer be triangular, but block triangular. All expected eigenvalues are in this case not contained in the diagonal but in the blocks on the diagonal, where each strongly connected component forms a block of minimum size two. Edges to nodes in a given connected component from other nodes will affect the expected eigenvalues of that component, which could affect the convergence rate. Note that $\boldsymbol{F}$ remains stochastic, so $\boldsymbol{F}$ will still have the unit eigenvalue.

References

[1] J. Potter, J. Alves, D. Green, G. Zappa, I. Nissen, K. McCoy, The JANUS underwater communications standard, in: 2014 Underwater Communications and Networking, UComms, 2014, pp. 1–4.

[2] Roberto Petroccia, João Alves, Giovanni Zappa, JANUS-based services for operationally relevant underwater applications, IEEE J. Ocean. Eng. 42 (4) (2017) 994–1006.

[3] Henry Dol, Koen Blom, Paul van Walree, Roald Otnes, Håvard Austad, Till Wiegand, Dimitri Sotnik, Adaptivity At the Physical Layer, Springer Berlin Heidelberg, Berlin, Heidelberg, 2020, pp. 13–40.

[4] Emil Wengle, John Potter, Damiano Varagnolo, Hefeng Dong, A JANUS-based consensus protocol for parametric modulation schemes, in: Proceedings of the 16th International Conference on Underwater Networks & Systems, WUWNet '22, Association for Computing Machinery, New York, NY, USA, 2022.

[5] Ivor Nissen, Frank Kramer, Bernhard Thalheim, Underwater cooperation and coordination of manned and unmanned platforms using S-BPM, in: Information Modelling and Knowledge Bases, vol. 30, IOS Press, 2019, pp. 137–146.

[6] Fatih Bülbül, Thies Petersen, Michael Recker, Fabian Sell, Kim-Fabian Wachlin, Ivor Nissen, Fehlertoleranz Bei Prozessabläufen: Mit Anwendungen Bei Akustischen Unterwassernetzwerken (Bachelor's thesis), Christian-Albrechts-Universität zu Kiel, Kiel, 2017.

[7] Roald Otnes, An underwater first contact method using JANUS, in: 2022 Sixth Underwater Communications and Networking Conference, UComms, 2022, pp. 1–5.

[8] Jing Yan, Ziqiang Xu, Yan Wan, Cailian Chen, Xiaoyuan Luo, Consensus estimation-based target localization in underwater acoustic sensor networks, Internat. J. Robust Nonlinear Control 27 (9) (2017) 1607–1627.

[9] Michael Kevin Maggs, Steven G. O'Keefe, David Victor Thiel, Consensus clock synchronization for wireless sensor networks, IEEE Sens. J. 12 (6) (2012) 2269–2277.

[10] Hao Chen, Huifang Chen, Ying Zhang, Wen Xu, Decentralized estimation of ocean current field using underwater acoustic sensor networks, J. Acoust. Soc. Am. 149 (5) (2021) 3106–3121, 05.

[11] Mohammad Mehdi Asadi, Amir Ajorlou, Amir G. Aghdam, Stephane Blouin, Global network connectivity assessment via local data exchange for underwater acoustic sensor networks, in: Proceedings of the 2013 Research in Adaptive and Convergent Systems, RACS '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 277–282.

[12] JANUS wiki, Online https://www.januswiki.com/tiki-index.php. (Accessed Feb 2024).

[13] Soummya Kar, José M.F. Moura, Distributed consensus algorithms in sensor networks: Quantized data and random link failures, IEEE Trans. Signal Process. 58 (3) (2010) 1383–1400.

[14] Tuncer Can Aysal, Mark J. Coates, Michael G. Rabbat, Distributed average consensus with dithered quantization, IEEE Trans. Signal Process. 56 (10) (2008) 4905–4918.

[15] Acoustic Research Laboratory, UnetStack3, National University of Singapore, and Subnero Pte Ltd, Online https://unetstack.net/. (Accessed 11 Sep 2023).

[16] Akshay Kashyap, Tamer Başar, R. Srikant, Quantized consensus, Automatica 43 (7) (2007) 1192–1203.

[17] C.J. Clopper, E.S. Pearson, The use of confidence or fiducial limits illustrated in the case of the binomial, Biometrika 26 (4) (1934) 404–413.

[18] Måns Thulin, The cost of using exact confidence intervals for a binomial proportion, Electron. J. Stat. 8 (1) (2014) 817–840.

[19] Matti Leppäranta, Kai Myrberg, Physical Oceanography of the Baltic Sea, Springer Science & Business Media, 2009.

[20] I.P. Chubarenko, N. Yu Demchenko, E.E. Esiukova, O.I. Lobchuk, K.V. Karmanov, V.A. Pilipchuk, I.A. Isachenko, A.F. Kuleshov, V. Ya Chugaevich, N.B. Stepanova, V.A. Krechik, A.V. Bagaev, Spring thermocline formation in the coastal zone of the southeastern Baltic Sea based on field data in 2010–2013, Oceanol. (Washington. 1965) 57 (5) (2017) 632–638.

[21] Paul A. van Walree, Propagation and scattering effects in underwater acoustic communication channels, IEEE J. Ocean. Eng. 38 (4) (2013) 614–631.

[22] Mohammad Sharif, Abolghasem Sadeghi-Niaraki, Ubiquitous sensor network simulation and emulation environments: A survey, J. Netw. Comput. Appl. 93 (2017) 150–181.

[23] Fabio Fagnani, Sandro Zampieri, Randomized consensus algorithms over large scale networks, IEEE J. Sel. Areas Commun. 26 (4) (2008) 634–649.

[24] Federica Garin, Luca Schenato, A Survey on Distributed Estimation and Control Applications using Linear Consensus Algorithms, Springer London, London, 2010, pp. 75–107.

[25] Y.H. Wang, On the number of successes in independent trials, Statist. Sinica 3 (2) (1993) 295–312.

Emil Wengle received the M.Sc. degree in engineering physics with specialisation in embedded systems from Uppsala University, Uppsala, Sweden, in 2020. He is currently pursuing a Ph.D. in underwater acoustic communication at the Norwegian University of Science and Technology, Norway.

Elias Strandell Erstorp received his M.Sc. in Naval Architecture in 2014 from the KTH Royal Institute of Technology in Stockholm. He began his doctoral studies in underwater communication and networking in 2017, following a few years of work with unmanned surface and underwater vehicles as a research engineer.

Viktor Lidström received an M.Sc. degree in space engineering, with a specialisation in spacecraft and instrumentation, from the Luleå University of Technology in 2017. He is currently pursuing a PhD-degree in underwater acoustic communication, with a focus on robust noncoherent links; other research interests include link adaptation and flooding networks.

Damiano Varagnolo received the Dr. Eng. degree in automation engineering and the Ph.D. degree in information engineering from the University of Padova respectively in 2005 and 2011. He worked as a research engineer at Tecnogamma S.p.A., Treviso, Italy during 2006–2007 and visited UC Berkeley as a scholar researcher in 2010. From March 2012 to December 2013 he worked as a post-doctoral scholar at KTH, Royal Institute of Technology, Stockholm. From January 2014 to November 2019 he worked first as Associate Senior Lecturer and then as Senior Lecturer at LTU, Luleå University of Technology in Sweden, teaching system identification and state-space based automatic control. He is now serving as Professor at NTNU in Trondheim within the Department of Engineering Cybernetics. His research interests include statistical learning, distributed optimisation, and distributed nonparametric estimation, with a special focus on applications including identification and control for the built environment, learning analytics, and muscular rehabilitation

Hefeng Dong received the B.Sc. and M.Sc. degrees in physics from the Northeast Normal University, China in 1983 and 1986, respectively, and the Ph. D. degree in geoacoustics from the Jilin University, China, in 1994.

From 1986 to 1994, she was a lecturer of physics with the Northeast Normal University, China where she was associate professor from 1995 to 2000. She was a visiting scholar and post doctorial fellow at the Norwegian University of Science and Technology, Norway between 1999 and 2001. From 2001 to 2002 she worked as a research scientist at the SINTEF Petroleum Research, Norway. Since 2002 she has been Professor in Acoustic Remote Sensing with the Norwegian University of Science and Technology, Norway. She was on sabbatical with the Underwater Acoustics Laboratory, University of Victoria, Canada, the College of Earth, Ocean and Environment, University of Delaware, USA, and the Laboratory of Mechanics and Acoustics, France in the periods of 2008–2009, 2014–2015, and 2019–2020, respectively. Her research interests include wave propagation, passive acoustics, geoacoustic modelling and inversion, signal processing in ocean acoustics and seismic, and underwater acoustic communication. Dr. Dong is a member of the Acoustical Society of America and IEEE.