Turk J Elec Engin, VOL.15, NO.2 2007, © TÜBİTAK

A Communication Module and TDMA Scheduling for a Swarm of Small Submarines

Ram SOMARAJU and Felix SCHILL
Research School of Inf. Sciences and Eng. Rsise Building 115
The Australian National University Canberra Act 0200 AUSTRALIA
e-mail: Somaraju@rsise.anu.edu.au

Abstract

Swarming algorithms usually require fast and reliable local information distribution among neighbouring nodes. Additionally, many applications also require global information distribution in the shortest possible time. We show that communication links with a range which is small compared to the size of the swarm offer many advantages. We propose a system using low frequency radio communication as a viable alternative to acoustic communication. A time division multiple access algorithm is presented together with results from simulations. The presented system can be effectively used in large scale swarms of small submersibles.

1. Introduction

Compared to traditional networks, communication in swarms of robots introduces a set of new paradigms. Most commercially available networks are designed for applications which favor point-to-point connections (unicast), and sporadically active nodes. Examples are wireless computer networks (i.e. WiFi 802.11) and mobile phone networks. However, in networks of swarming (identical) robots, communication is continuous - nodes have to constantly provide updates. Additionally it is favoured to distribute all available local information among as many nodes as possible, locally first and eventually globally. This communication mode is called *Omnicast* (a.k.a. global gossiping) [1]. Example applications for *Omnicast* are distributed maximum/minimum calculations, averaging, determining the center of gravity of the swarm, or gradient search on the sensor data of interest.

Traditionally underwater communication relied on acoustic waves. Acoustic channels severely suffer from multipath propagation and high latency, which is less stringent in radio channels. However, commonly used radio links use typically high frequencies (GHz range), and these are heavily attenuated in water due to conductivity. Low radio frequencies are less affected by attenuation, and offer a suitable alternative for short range communication with acceptable power consumption and latency, but introduce limitations on the available bandwidth for data communication.

The short range implies that large scale networks are multi-hop wireless networks. It also implies that the channel can be space-multiplexed between participants sufficiently far apart. We will show that in terms of local and global information distribution in swarms, limited range radio links are actually an advantage.

We therefore designed a communication module based on radio waves. Due to the constraints imposed by water, careful design is required to develop a radio communications module. As explained in section 3

Figure 1. The *Serafina Mk II* prototype autonomous underwater vehicle

Gaussian minimum shift keying (GMSK) is ideally suited for the underwater channel and the hardware required to implement this type of modulation is described. The complete system will be used in a swarm of 50 cm long *Serafina* miniature submarines. The small size has many advantages in terms of handling and usability, but it does impose constraints on module size and power consumption.

Finally, robotic swarms require low, predictable latency for distributed control, and robustness against changing network topologies. It might be possible to ignore these constraints in some land-based applications due to the availability of high bandwidth radio links. In the underwater environment, the limited resources require a channel access protocol which is optimised for the specific requirements. The sum of constraints point towards Time Division Multiple Access protocols (TDMA). A suitable algorithm will be presented in section 4. The following section will highlight some fundamental properties of the scheduling problem.

2. The Optimal Omnicast problem

For theoretical considerations it is assumed that nodes can collect all information they have received and redistribute it in a single message. Note that this is different from the analysis of broadcast capacity presented in [2]. The broadcast capacity of wireless networks obtainable by each node is limited by $O(C/n)$ (C is the channel capacity, n the number of nodes). This assumes full relaying of the original messages to the entire network. However, if nodes can collate and compress the information they collected so far, this limitation does not necessarily apply. The analysis in this paper assumes that all collected information can be packed into a single message. There are applications where this assumption can be easily met. In practical scenarios, it is nat always necessary to relay every single message in full. A simple example is the distributed calculation of a maximum - only a single value (the current local maximum) represents all information collected so far. Swarm control algorithms often exchange local parameters such as swarm density, preferred direction and velocity. These parameters intrinsically contain information contributed by many or all vehicles, and have to be distributed among the entire swarm. A communication system that implements omnicast ensures that information is gathered and redistributed globally, regardless of the content of the actual messages which

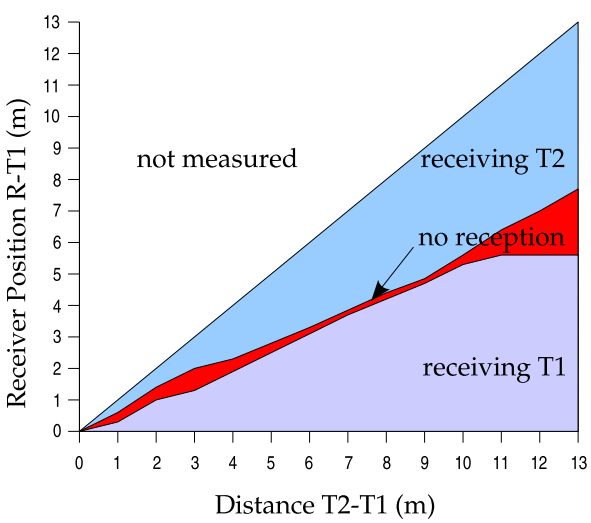


Figure 2. Jamming experiment with two transmitters and one receiver

are exchanged.

Communication between nodes is assumed to occur in discrete, synchronised time steps. The required clock synchronisation between nodes can be easily achieved in practice with common methods.

2.1. Network model

A common way to model communication networks is using a graph. Let $G = (V, E)$ be a graph describing a network with $n = |V|$ nodes. Vertices $v \in V$ represent communication nodes, containing a transmitter, a receiver and a processing unit. A directed edge $e \in E \subset V \times V; e = (u, v); u, v \in V$ expresses that node u can (in principle) send data to node v. For completeness, this includes reflective edges $(u, u) \in E \; \forall u \in V$. A node $v \in V$ receives a message if and only if there exists exactly one node $u \in V$, so that u is transmitting and $(u, v) \in E$. If there exists more than one node with these properties, a collision occurs at v. In case of a collision, node v can in general not decode the message or detect the collision, meaning it can not distinguish between a collision and noise.

It is often assumed for simplicity, that the network graph is symmetrical and connected. This assumption can be met if all nodes are identical, and have therefore an identical range.

It has to be noted that this network model, though common in the literature, does not reflect the reality as found in the experiments with phase-modulated signals (refer to figure 2 and [3]). Collisions only occur if the strongest competing signals at a node have comparable signal strength (figure 2). Generally, a node will successfully receive the message of the closest transmitting node in range, provided the signal to noise ratio of the strongest signal over noise plus all competing signals is good enough. The graph model is more conservative, which means that a system might perform better in reality than predicted by the common network model. Respectively, a system which exploits the more accurate collision model can perform better than a system optimised for the graph-theoretical model. For the theoretical analysis in this section the described graph model is assumed.

Definition 2.1 (Omnicast) *Let $G = (V, E)$ be a graph describing a communication network with $n = |V|$ nodes. In the start state, every node $u \in V$ has a set $I_u(t_0)$ of information tokens, which contains exactly one unique token B_u of information. During the communication phase, a node v updates its set $I_v(t+1) = I_v(t) \cup I_w(t)$, if and only if it successfully receives a message from $w \in V$ in time step t (refer to the network model for message exchange), and $I_v(t+1) = I_v(t)$ otherwise. The end state is reached, when all nodes have the full set with all tokens, $I_u(t) = \{B_v | v \in V\}$ for all $u \in V$.*

Having defined the task to solve, we can now define an optimality problem:

Definition 2.2 (The Optimal Omnicast Problem) *Find a schedule $S_G = (T_1, ..., T_t), T_i \subset V$ for $i = 1...t$, with T_i being the set of sending nodes in time step i, such that S_G solves the omnicast on the network graph G, and t is minimal.*

2.2. Discussion

The Optimal Omnicast is NP-hard [4]. Upper and lower bounds and some special cases are described in [1]. For symmetric graphs an upper bound is $2n - 2$ time steps. A different upper bound of $O(\Delta(G) \cdot d(G))$ is given in [4], with Δ being the maximum degree and d the diameter. These bounds are not comparable - it depends on the graph which one is lower.

The upper bound of $2n - 2$ steps can be extended to the more general case of directed, strongly connected graphs.

Theorem 2.3 *Let G be a directed, strongly connected graph (there exists a path from each vertex to every other vertex) with n nodes, such that an omnicast solution exists. Omnicast can be solved on G in at most $2n - 2$ steps.*

Proof It is possible to split the omnicast problem into a convergecast followed by a broadcast. For simplicity we assume that only one node sends at a time. To obtain the schedule for the convergecast, select a target node. Invert all edges, and perform a breadth-first search, starting from the target node. The nodes send in the reverse order in which they were discovered by the breadth-first search (not including the target node). Similarly, to obtain the schedule for the broadcast, perform a breadth-first search starting from the target node, with the original edge orientation. The nodes send in the order in which they were discovered (including and starting with the target node).

The convergecast requires at most n-1 steps (the target node does not have to send). The broadcast requires at most n-1 steps (the nodes discovered in the last round of the breadth-first search do not have to send - this is at least one node). After the convergecast finishes, the target node has all information. After the broadcast, every node has all the information. Therefore, omnicast can be solved in at most 2n-2 steps. □

It is obvious that there are better solutions. The broadcast will usually require much less than n-1 steps, depending on the number of nodes discovered in the last round. Also, by applying concurrent schedules, omnicast converges much faster in most cases. Special cases like fully connected graphs, or graphs of maximum degree 2 have solutions with n time steps [1]. Furthermore, an exhaustive search revealed that there are solutions for all connected symmetric graphs up to size 6 with at most n time steps. For $n = 6$ the majority of graphs only requires 5 steps. For larger graphs between time steps required over the number of nodes typically becomes even smaller.

An approximated upper bound can be derived geometrically, to gain more insight into the performance of networks embedded in 2d or 3d space. Let $G = (V, E)$ be the network graph of a network, and P the longest shortest path in G (diameter $d = |P|$). Select every fourth node on P: $K := P_2, P_6, P_{10}, ...$, so that their two-hop neighbourhoods $k \in K \mapsto C(k) := \{n \in V : |n, k| \leq 2\}$ are overlapping. There are $O(d(G)/4)$ nodes in K. Each cell $C(k)$ can achieve complete information exchange among all nodes in $C(k)$ in less than 2γ time steps, with γ being the size of the largest 2-hop neighbourhood (every node in a 2-hop neighbourhood sends twice in a dense schedule). To transmit all information along P this has to be repeated $d/4$ times (all other paths are shorter, therefore omnicast is complete). The upper bound is therefore $U = O(\frac{1}{2}\gamma \cdot d)$ time steps.

Assume a homogenous network of circular shape with n nodes in a plane with equal node density, network area A, and r being the communication range. The graph degree can be approximated by $\Delta' = O\left(n\frac{r^2}{A}\right)$. The network diameter is approximately $d' = O\left(\sqrt{A}/r\right)$. The size of a 2-hop neighbourhood is approximately $\gamma' = O\left(n\frac{4r^2}{A}\right)$. It follows that the geometrically approximated upper bound is $U' = \gamma' \cdot d' = O\left(n\frac{r}{\sqrt{A}}\right)$. The 3D case is very similar, and leads to an upper bound of $O\left(n\frac{r^2}{\sqrt[3]{A}}\right)$. This geometric approximation is only valid for large n, and if r is greater than the maximum distance between nodes. Obviously the connectivity of the network suffers with small values for r, and the network can become disconnected if the value is too small. What can be seen is that, at least for sufficiently large networks and carefully chosen communication range, the upper bound is proportional to the communication range r for a given network size. It is obvious that this geometric approximation does not apply for small networks. Simulation results presented later give actual numbers for various network sizes, and provide support for the presented approximation.

The upper bound U' suggests that r can be chosen for a given network area and density, so that the upper bound for omnicast is smaller than n. In fact, solutions can be found that require less than n time steps for networks with small degree. However, fully connected networks have a lower bound of n steps. Additionally, fully connected graphs suffer from poor local update rates - the frequency at which nodes can send is exactly $1/n$. Networks with lower connectivity allow for parallel communication in different parts of the network; the per node transmission frequency is proportional to $1/d(G)$. Fast local update rates are important for swarm control.

Generally this means that for a given swarm size, links with a range smaller than the swarm diameter are a significant advantage. In a swarm with evenly spaced robots, the communication links optimally connect each node only to its direct geometric neighbours. However, in real swarms it can be necessary to increase the range for better redundancy.

3. Communication module

Traditionally submarines have relied on acoustic waves for underwater communication. However, acoustic waves in water have large propagation distances which implies that the links are long range. For instance, at a carrier frequency of $30kHz$, the waves are attenuated by only $0.3db/m$ [5]. As explained in the previous section this is not ideal for small submarines in a swarm. Further, acoustic communication is affected by multi-path propagation which makes it difficult to decode the transmitted signal even for high received signal to noise ratios [6,7]. Therefore, if two nodes in a swarm are separated by a large distance then the signals

transmitted by one will reach the second one but cannot easily be decoded. Hence, two nodes that are not connected with each other might still interfere with each other causing a collision and will therefore disrupt the scheduling algorithm. Moreover, because the velocity of acoustic waves is several orders of magnitude lower than that of electromagnetic waves, acoustic communication suffers from large latencies. Finally, acoustic communication requires large modems which might be a problem for submarines in a swarm that are designed to carry only small payloads.

Longwave radio communication is a possible alternative to acoustic communication. However, if the wavelength is very long then big antennas would be required which is again a problem for submarines that carry small payloads. Therefore a high carrier frequency needs to be used for swarm communication. However, due to the conductivity of saline water high frequency radio waves are severely attenuated and the carrier frequency needs to be carefully chosen. In what follows we assume plane wave propagation and derive relations for rate of attenuation and phase shift as a function of frequency for different salinities.[1] Using this model we design a communication module and describe a possible hardware implementation of this module.

3.1. Simulation

The permittivity of water is a function of its temperature, salinity and frequency. A formula for this permittivity variation was derived in [12] which is restated here for convenience. The relative permittivity $\epsilon_r(\omega, T, S)$ at temperature T, salinity S and frequency ω is

$$\begin{aligned}\epsilon_r(\omega, T, S) = \epsilon_\infty(T) + \frac{\epsilon_s(T)(1-\alpha(T)S) - \epsilon_1(T)}{1 + j\omega\tau_1(T)} \\ + \frac{\epsilon_1(T) - \epsilon_\infty(T)}{1 + j\omega\tau_2(T)} \quad - \frac{c(T,S)}{\epsilon_0\omega^2(1 + j\omega^{eff}(T,S)/\omega)}\end{aligned} \tag{1}$$

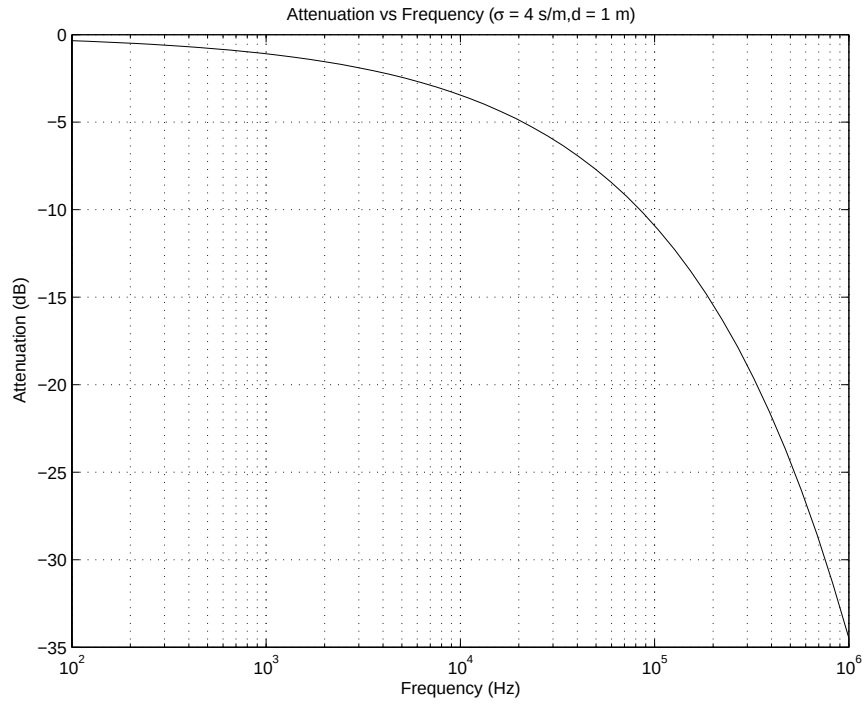
Here, $\epsilon_s(T)$ is the static relative permittivity, $\epsilon_\infty(T)$ is the permittivity at infinite frequency and τ_1 and τ_2 are Debye relaxation times for fresh water. $\epsilon_1(T)$ is the relative permittivity that accounts for a second relaxation process of fresh water. Also, $\alpha(T) = 0.00314ppt^{-1}$ and $C(T,S) = 1\times10^12\cdot S\mathrm{C}^2/\mathrm{kg}$ are parameters determined from experimental measurements [12]. The rate of attenuation, A and wavelength, λ for plane waves can now be calculated using the dielectric constant:

$$\lambda = \frac{2\pi c}{\omega\Re\{\sqrt{\epsilon}\}} \tag{2}$$

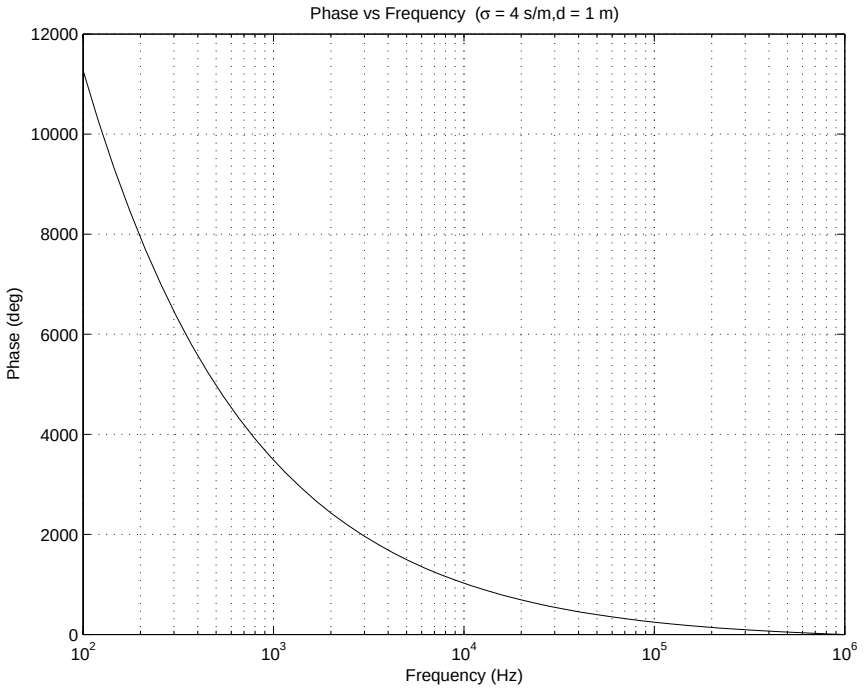
$$A = \exp\{-2\pi\tan(\angle\epsilon/2)\} \tag{3}$$

Here, $\epsilon = \epsilon_r\epsilon_0$ is the permittivity of water and $\angle\epsilon$ is the argument of ϵ. Using this formula, the velocity of propagation and rate of attenuation for plane waves can be determined as a function of frequency, temperatur and salinity. This can then be used to approximately predict the channel's frequency response. The rate of attenuation and phase shift for seawater, with conductivity $\sigma = 4S/m$ at a temperature of 25 degress is shown in figure 3. These plots may be used to calculate the highest possible propagation frequency that may be used based on the propagation distance and receiver specifications. For instance if a propagation distance of 5m is required in seawater and the transmitter power is 1dbm and the required signal strength at the receiver is -100dbm then the maximum carrier frequency is 150kHz.

[1] It is possible to use more complex models that more accurately model wave propagation (e.g. see [8–11]) but this was not considered here.



(a)



(b)

Figure 3. Attenuation and Phase shift v Frequency for plane wave propagation in seawater

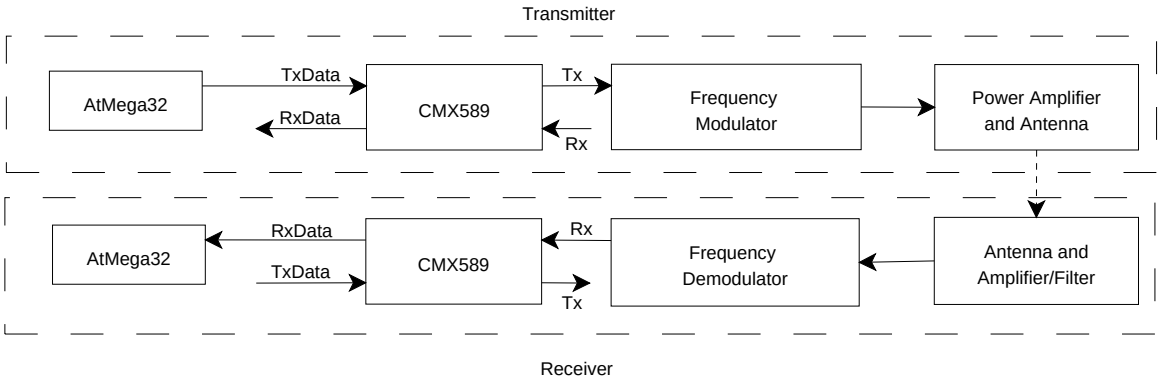


Figure 4. Block Diagram of communications module

3.2. Module Description

Two important points emerge from the graphs of attenuation and phase shift versus frequency: firstly, due to severe attenuation of electromagnetic waves, the received signal to noise ratio will be very low. This is compounded by the fact that these submarines will have limited power. Secondly, the phase delay is a nonlinear function of frequency. This leads to severe distortion when using wide-band signals. This will particulary be the case if a more accurate wave propagation model is considered.

To overcome the low signal to noise ratio it was decided that frequency modulation be used. However, frequency shift keying has high bandwidth requirements and therefore it was decided that Gaussian Minimum Shift Keying (GMSK) modulation be used. This type of modulation with a BT value of 0.3 offers extremely narrow bandwidth operation [13]. Unfortunately, because low frequencies need to be used for communication in both swimming pool and sea water none of the commercially available chips are appropriate for underwater communication. We therefore designed the GMSK modem shown in figure 4. This modem is currently being implemented and will be tested in the near future. For the remainder of this section we discuss the design issues for the transmitter and receiver.

3.3. Transmitter Design

None of the commercially available pass band GMSK modulator chips are suitable to use in salt water because a very low carrier frequency is required. Therefore it was decided that a baseband GMSK chip, CMX589 be used in conjunction with a frequency modulator to develop a pass band GMSK modulator. An Atmel processor AtMega32 is used to control the whole transmitter circuit and can also be used to interface with an external processor through an RS232 connection. In the remainder of this subsection I will discuss the design issues of the transmitter circuit.

3.3.1. Frequency modulator

For GMSK modulation, data patterns consisting of long strings of ones or zeros have a spectral response that extends down to DC [14]. This is the most important characteristic that determines the type of frequency modulator to be used because most audio frequency modulators have a response extending down to a few hundred Hertz. There are two options that can be used to design a frequency modulator with response extending to DC.

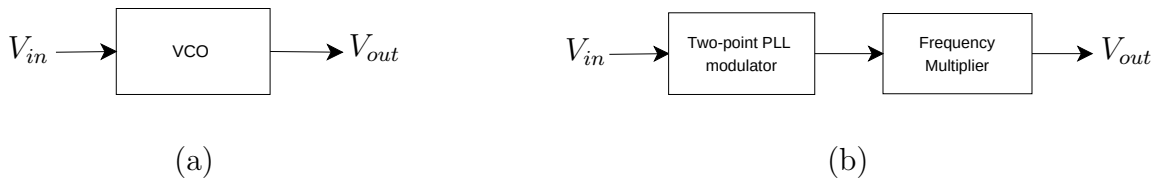


Figure 5. a) VCO modulator b) Two-point modulation with frequency multiplier

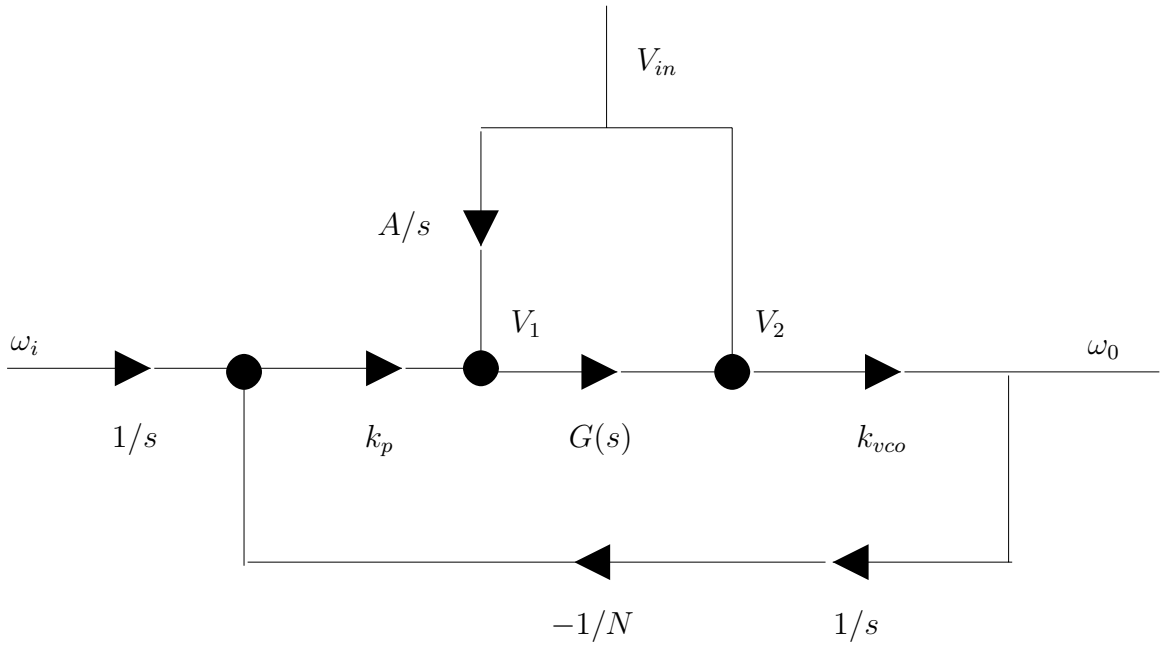


Figure 6. Two point PLL modulator

Firstly, one can use a voltage controlled oscillator (VCO). In this type of modulation, the modulating signal is simply fed into a VCO as shown in figure 5. Here, V_{in} is the modulating signal and V_{out} is the frequency modulated output. This design is flawed because the center frequency of the VCO drifts considerably with both time and temperature and is not viable in the long-term.

A better alternative consists of using phase-frequency modulation as explained in [15]. Figure 6 shows an equivalent block diagram of this type of modulation. The modulator consists of a phase-lock loop (PLL) with the modulating voltage input into the PLL at two distinct points. In this figure, k_{vco} is the gain of the voltage controlled oscillator (VCO), k_p is the gain of the phase detector, $G(s)$ is the transfer function of the low pass filter, $1/N$ represents the gain of the clock divider and A/s is the integrator with gain A. V_{in} is the modulating voltage, ω_o is the frequency of the modulated waveform and ω_i is the carrier frequency. It

is easy to show that

$$\frac{\omega_0}{V_2} = \frac{sk_{vco}}{s + k_p k_{vco} G(s)/N} \quad (4)$$

$$\frac{\omega_0}{V_1} = \frac{sk_{vco} G(s)}{s + k_p k_{vco} G(s)/N} \quad (5)$$

Therefore,

$$\frac{\omega_0}{V_{in}} = k_{vco} \frac{AG(s) + s}{s + k_p k_{vco} G(s)/N}. \quad (6)$$

Hence, if $A = k_p k_{vco}/N$, then

$$\frac{\omega_0}{V_{in}} = k_{vco}. \quad (7)$$

This is the ideal flat frequency response to d.c. that is required.

However, if we assume that the carrier frequency is $150kHz$ and assume that V_{in} is a $1Vp-p$ signal then $k_{vco} \approx 75000$. This implies that the gain A of the integrator, which is proportional to K_{vco} is very high. In fact, even if there is a small offset voltage of a few millivolts at the input of the integrator, which is always the case with real op amps, the output of the integrator will saturate at one of the rail voltages.

Hence, a low center frequency is used to do the frequency modulation. Then, the modulated signal is frequency multiplied to the required carrier frequency and this will ensure that the required deviation is achieved as the deviation is also multiplied by the same amount (see fig 5).

3.3.2. Power Amplifier and Antenna

It was decided that a loop antenna be used because they work much better underwater than linear antennas [9]. However, the small size of the submersibles forces the usage of small loop antennas whose length is much smaller than a wavelength. These small antennas are essentially inductive. Therefore to maximise the total energy radiated by the antenna, one must use a tuning capacitor in series with the antenna. The antenna radiation resistance increases with increasing number of turns and so does the inductance. Therefore to ensure the maximum possible radiation resistance, the tuning capacitor must have a very small value. The value chosen was $47pF$ which is small but not too small when compared to the stray capacitance. Once this capacitance is determined one can choose the inductance of the antenna because the carrier frequency $f_0 = 1/2\pi\sqrt{LC}$. This antenna/capacitor tuned circuit is driven by the frequency modulator using the mosfet driver TC4427.

3.4. Receiver Design

Similar to the transmitter the baseband GMSK modem CMX589, is used to demodulate a baseband GMSK signal. The passband signal is converted to a baseband signal using a frequency demodulator which is described below. Again, an Atmel processor AtMega32 is used to control the receiver.

3.4.1. Antenna, Amplifier and Filter Circuit

Similar to the transmitting antenna, the optimal antenna subject to the physical constraint of being housed in a small submersible is a small loop antenna. This loop antenna is connected to a parallel RC circuit to

tune it to the required carrier frequency while ensuring that the Q value of the circuit is not too big. The output of the tuned circuit is fed into a JFET pre-amplifier described in [16, Figure]. A JFET pre-amplifier is used so as to not load the tuning circuit. The output of this pre-amplifier is fed into the SA605 Phillips semiconductor chip which acts as a bandpass amplifier and provides a gain of $105dB$ at the carrier frequency.

3.4.2. Frequency Demodulator

A PLL demodulator offers a slight improvement (~ 2dB) over a simple limiter discriminator at low signal to noise ratios [17]. Hence, it was decided that a phase-lock loop demodulator be designed. Again, two options are available in designing this type of demodulator. Firstly, a two-point method similar to the one used for the modulator may be used [15]. This method has the advantage of having a flat frequency response independent of the loop bandwidth. This design is essential if large bit-rates are required; particularly because of the low carrier frequency. For instance, if a second order loop is used, then the loop bandwidth must be at least 30-100 times lower than the carrier frequency and also be greater than the bit rate. However, this method requires more components and it was decided that following easier to implement alternative be used.

We use a third-order PLL to design the frequency demodulator. Also, it is essential to use at least a type 2 loop to counter the doppler shift that might occur as a result of the constant motion/acceleration of the submersibles. Therefore, the transfer function of the low pass filter used in PLL of this demodulator is [15]:

$$G(s) = \frac{1}{s\tau_1}\left[\frac{1+s\tau_3}{1+s\tau_2}\right]$$

Here,

$$\begin{aligned}\tau_1 &= \frac{\tan\phi + \sec\phi}{\omega_n}\\ \tau_2 &= \frac{1}{\omega_n(\tan\phi+\sec\phi)}\\ \tau_3 &= \frac{k}{\omega_n^2}(\tan\phi+\sec\phi)\end{aligned}$$

Here, ω_n is the loop bandwidth and it must be at least 1.5 times the bit rate [18], $\phi = \pi/3$ rad is the required phase margin and k is the loop gain. This transfer function can be easily implemented using a simple active filter [15]. A block diagram of this PLL is shown in figure 7. Here, ω_{in} is the frequency of the modulated signal and V_{out} is the demodulated waveform. Also, k_p, k_{vco} and $1/N$ are the gains of the phase detector, VCO and clock divider respectively. $G(s)$ is the loop filter and $H(s)$ is a fourth-order low-pass Butterworth filter that filters out the high frequency harmonics. We will now describe a distributed scheduling algorithm for TDMA scheduling.

4. Distributed scheduling for channel access

The communication module presented in the previous section offers a single half-duplex channel. The problem of multiple channel access has to be addressed before the radio system can be used in swarming applications. Swarming and robotic control impose real time constraints. Quick updates are important, and also the

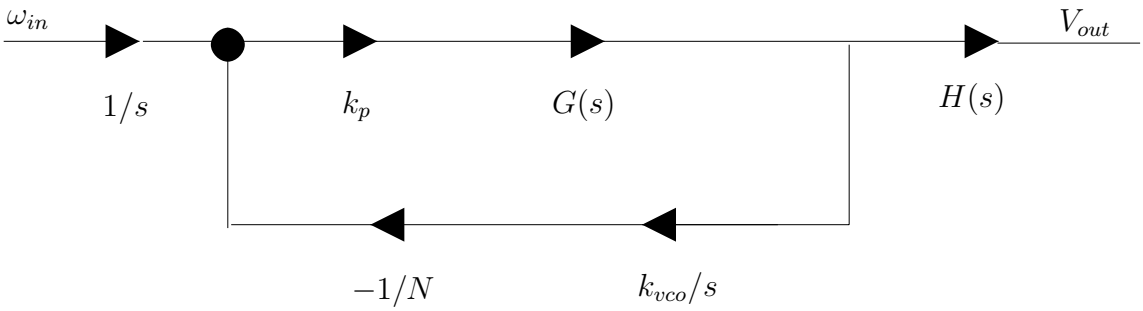


Figure 7. Phase-Lock Loop demodulator

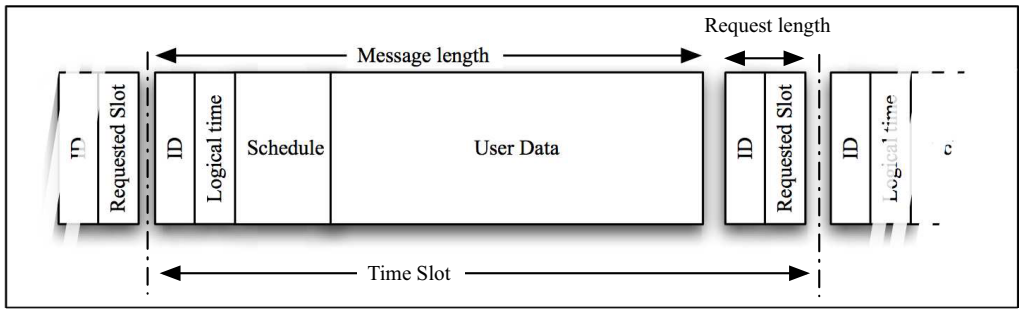


Figure 8. Format of the data packet and the time slot

predictability of the communication system. Consistent update frequencies and known latencies allow the swarm control simpler and more robust. Under the given constraints, deterministic Time Division Multiple Access (TDMA) has significant advantages over other channel access methods.

The Distributed Dynamical Omnicast Routing (DDOR) algorithm presented here is a TDMA scheduling algorithm which is geared towards high throughput and quick global information exchange. The variant of DDOR presented here is based on an algorithm which has been published earlier [19]. The version described here is slightly simplified, and has improved performance and shorter startup time. The DDOR algorithm provides low level channel access. Specific routing requirements are not considered, and have to be implemented on higher network layers. The goal of DDOR is to maximise utilisation of available capacity at all times, and fairly distribute this capacity among all nodes.

4.1. The Basics

DDOR is a distributed algorithm for TDMA scheduling in wireless multi-hop networks. It is computationally inexpensive, and can be implemented on a small microcontroller. It is designed for continuous traffic and quick local and global information dissemination.

Communication nodes maintain a logical clock $L(t)$, an integer clock which increments at the beginning of each time slot. The logical clocks are synchronised during the startup of the network by monitoring incoming messages. It can be assumed that the logical clock $L(t)$ is equal and in sync up to sufficient accuracy on all participating nodes (this has been tested and verified both in simulations and on real hardware).

To allow for dense packing of schedules on one hand, but reconfigurability on the other hand, time slots are divided into packet slots, separated by short 2-byte request slots. Figure 8 shows the packet and time slot structure. Each packet contains the unique identifier of the sender, the sender's logical clock, and the current local schedule of the sender. A request packet contains the sender's ID and the number of the requested schedule slot.

The following data structures are maintained by each node:

- The visible neighbourhood N. This is database of all nodes that are within range of the node. The node database is maintained according to received messages. It is assumed that a node which message has been received is within range. Nodes are removed from the list and the schedule if no messages from them were received over a certain amount of time (at least two schedule rounds). With every node in the list the node's local schedule is stored as received in the last message of this node. Removal of a node from the database also invokes the removal of that node from all locally stored schedules (the locally stored copies of schedules received from other neighboured nodes).
- The local schedule s_i. Nodes try to establish local collision-free schedules. The schedule of each node consists of a number of schedule slots. A slot can be either empty, blocked, or used by a node present in the visible neighbourhood. The local schedule is recalculated with every received message from the most recent schedules received from all nodes in visible neighbourhood.

A node can be in one of three states: *Listening*, *Requesting*, and *Running*. The default is *Listening*. The node remains in the *Listening* state for a random number of time slots, while updating its local schedule and visible neighbourhood list from received messages. If the visible neighbourhood list is still empty after the time out (no messages received), the node adds itself to the first slot of its local schedule and transmits a message containing this schedule. Otherwise the node sends out a request for the first empty slot in the current schedule, and returns to *Listening* for a random time. The requested slot (*own slot*) is stored locally for future reference.

4.2. The DDOR scheduling algorithm

All nodes keep track of their neighbourhood (all nodes from which they recently received a message). Nodes also save the most recent schedules they received from their neighbours. A neighboured node is marked as *established* if a complete message with schedule has been received from this node, and as *not established* if only a request has been received up to now (respectively if the last message from this node was a request).

The algorithm consists of two tasks, the transmitter task and the receiver task. The two tasks communicate indirectly through the node database, which has to be implemented as a multi-tasking save entity (i.e., a protected object).

4.2.1. Receiver task

The receiver task takes care of incoming messages. It only becomes active if a message is received, and therefore can also be implemented as an interrupt handler. This is especially useful for implementations on microcontrollers, on which a distributed programming language might not be available. This is the description of the receiver task in pseudocode:

```
loop:
   Wait_For_New_Message;
```

```
   Message:=Retrieve_Message;
   Synchronise_Clock(Message.Logical_Clock);

   If type of Message is 'Request' then
      If Message.Sender is in Node_Database,
         remove Message.Sender from Node_Database;
      end if;
      Clear requested slot from all schedules in Node_Database;
      Create new entry for Node_Database:
        Node.ID := Message.Sender;
        Node.Schedule := Empty_Schedule;
        Mark requested slot in Node.Schedule as used by Message.Sender
        Node.Established := false;
      Store new Entry 'Node' in Node_Database.
   else if type of Message is "Message" then
      update Node_Database with
       (Message.Sender, Message.Schedule, Established := True;)
   end if;
end loop;
```

4.2.2. Transmitter

The transmitter task is more complex than the receiver task. This is a periodic task, which becomes active at every beginning of a time slot or a request slot.

```
loop:
   Wait_For_Next_Time_Slot;
   Local_Schedule := Recalculate_Local_Schedule(Own_Slot);
   Current_Time_Slot := Calculate_Active_Time_Slot;
   If This_Node is in Local_Schedule, then
      State:=Run;
   else if not State=Listen then
      State:=Listen;
      Set Listen_Time to random number of time slots
   end if;

   Case State is
   Run:
      If Local_Schedule(Current_Time_Slot) = This_Node then
         If better slot with lower index available then
            with Probability of 33%:
               Prepare_Request for better slot;
               State := Request;
         end if;
         with Probability of 95%
            Transmit_Message(Local_Schedule);
      end if;
   Listen:
      If Listen_Time=0, then
         If Node_Database is empty, then initiate:
```

```
            Own_Slot := first empty schedule slot;
            Local_Schedule(Own_Slot) := This_Node;
            Transmit_Message(Local_Schedule);
         else
            Prepare Request for first empty slot in local schedule;
            State:=Request;
            Set Listen_Time to random number of time slots
         end if;
      else
         Decrement Listen_Time
      end if;
   end case;

   Wait_For_Next_Request_Slot;
   if State = Request then
      Clear this node from local Node_Database
       and all locally stored schedules;
      Send out Request for prepared slot;
      Set Own_Slot to Requested slot
   end if;
end loop;
```

The two crucial functions called in the transmitter task are *Recalculate Local Schedule* and *Calculate Active Time Slot*. The nodes in the local node database are denoted $n_j \in N$; each node in the local database has a locally stored schedule s_j (the most recently received schedule from each neighbour) with schedule slots $s_{i,j}$. Schedules of nodes from which no schedule has been received yet are empty, i.e. all slots of that schedule are marked as empty slots. The calculation of the local schedule is described here by the function $s_i(o)$:

$$s_i(o) := \begin{cases} b: & \exists n_j \in N : s_{i,j} \neq e \wedge s_{i,j} \neq b \wedge s_{i,j} \notin N \\ j: & j = \min\{A_i\} \\ o: & i = o \wedge \exists n_j \in N : s_{i,j} = o \wedge \forall n_j \in N : (s_{i,j} = o \vee s_{i,j} = e) \\ e: & \text{otherwise} \end{cases} \tag{8}$$

with $A_i := \{k : n_k \in N \wedge s_{i,k} = k\}$

It can be seen from the definition of $s_i(o)$ that collisions between competing nodes are resolved by favouring the node with the lowest index. This is an invariant which can be equally computed by all affected nodes. The "own" slot (the slot that a particular node requested last) is only assigned if at least one node in the 1-hop neighbourhood confirms this, and if the slot is otherwise unused.

The active time slot is calculated recursively from the current logical time t, and the current local schedule $s = s_1..s_l$. The schedule length l is assumed to be a power of 2.

$$\begin{aligned} \alpha : (N, S, N) &\mapsto N \\ (L(t), s_t, l) &\mapsto \alpha(L(t), s_t, l) := \begin{cases} 1 + L(t) \bmod l & | \quad s_{1+L(t) \bmod l} \neq e \\ \alpha(L(t), s_t, l/2) & | \quad s_{1+L(t) \bmod l} = e \\ 1 & | \quad l \leq 1 \end{cases} \end{aligned} \tag{9}$$

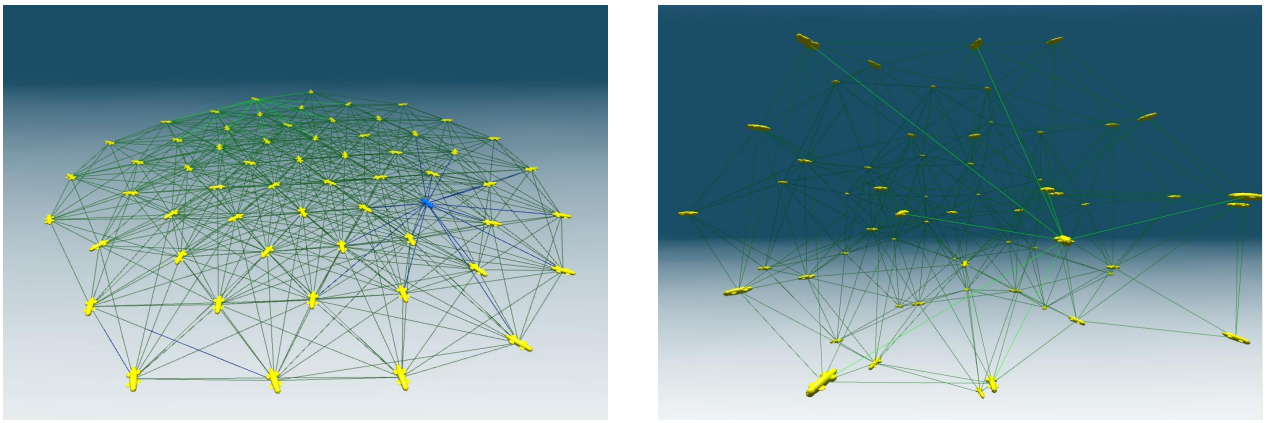


Figure 9. Simulation of a swarm of submarines in 2D configuration (left) and 3D configuration (right). The lines between submarines indicate that they are within communication range.

The initial call of the recursive function uses the maximal schedule length (must be a power of 2) as a parameter. The described mapping function has the advantage that it does not return empty time slots if the first slot of the schedule is filled, thus increasing utilisation especially in the case of sparse schedules. Alternatively other time slot mapping functions can be used.

4.3. Swarm Simulation

The distributed TDMA scheduling algorithm has been implemented both in a simulation environment and on a hardware system. Up to now the hardware system could only be tested with up to four nodes, which could indicate the principal similarity between simulation and reality, but doesn't allow in-depth analysis of the scheduling at larger scales. For small networks with size 4, the algorithm behaved identically in simulation and reality for all possible configurations.

A simulation environment has been implemented in Ada 2005, which provides a 3D arena for simulated submersibles. The simulation runs in realtime, and is multi-threaded. Each submarine consists of several tasks, and encapsulated data structures. Access to simulation parameters and other submarines is only possible over simulated sensory equipment. The simulation implements encapsulation, i.e. the access routines to simulated hardware are identical to the access routines to the actual, real hardware. This means that the tested algorithms can be directly ported to hardware without any changes to the implementation, and the algorithms should not experience any principal differences between simulation and real world. However, no simulation can fully emulate the real world, and there will always be subtle differences.

The submersibles are subject to a simplified force-based dynamics simulation. They can perform distance and bearing measurements on all their neighbours within their sensing range (set to 4 meters for the following experiments) at an update rate of 2 Hz. This allows the submarine to apply a primitive swarming rule, using simulated springs with a positive neutral length. The swarm will quickly converge towards a configuration with approximately equal distances between neighboured submarines, which leads to a triangular arrangement, if submarines are restricted to a plane, and a tetrahedrical configuration in the 3D case (Figure 9). It has to be noted that submarines try to equalise the distance between *all* their neighbours within their sensing range, which means that the minimum distance (and the average distance to closest neighbours) can be smaller than the length of the virtual spring. While not critical for the following evaluation of the communication system, this has to be kept in mind when considering the connection

between measured average distance of submarines within sensing range, and the average degree (or number of neighbours) of the network.

To simulate the communication system, the range and collision model has been implemented to closely match the behaviour found in the experiments (figure 2). Transmission durations are simulated in realtime, and message delivery is performed by the simulation arena according to the modelled link distance and competing messages, to detect possible collisions. While not being a physical simulation of wave propagation, it closely reproduces the results in message delivery found in physical experiments with previously designed and tested 122kHz digital longwave radio modules [3].

The following experiments were done running the dynamics simulation, the simple swarming rules, and the distributed scheduling algorithm (DDOR). The swarm is initialised with a virtual spring length of 1.5 meters. The sensing and communication range is set to 4 meters. After startup, the submarines quickly arrange themselves in a round disk-shaped (2D) or ball shaped (3D) configuration, equally spaced in a triangular configuration. Due to the ratio of average distance between subs and the maximum communication range this results in a fully connected network for all presented networks up to 60 nodes.

The communication network immediately starts building up local schedules, and after less than one minute, all submarines are included in the schedule and exchange information. After the initial stabilisation phase, the length of the virtual spring applied in the swarming rules is dynamically changed in various patterns. This causes the swarm to slowly expand or shrink, and the network to become less or more densely connected. The scheduling algorithm and the swarming behaviour are never stopped for the whole duration of the experiment. During this process, the following parameters are measured:

- Average distance. This is the average distance between a submarine and all submarines within its sensing range, averaged over all submarines. This value is closely linked to the virtual spring length.

- Average degree. This is the number of visible neighbours (submarines within communication range), averaged over all submarines.

- Omnicast Performance. This number indicates the required number of time slots to achieve *Omnicast* (full information exchange between all nodes). In contrast to the static case described in the definition of *Omnicast*, this number is calculated in a distributed way as follows. Every submarine maintains a list of counters $V = v_1, ..., v_n$ for all submarines in the swarm, which is sent out with every message. With every received message (containing the external counter list $x_1...x_n$), every local entry v_i is updated by $v_i(t+1) = \max(v_i(t), x_i)$. If all entries $v_1...v_n$ are equal, the local own counter is incremented by 1. The time between two counter increments reflects the average duration of omnicast in the current schedule, since every node has to receive every other node's update to be able to increment its own counter.

A sample run can be seen in figure 10. The plot shows the real time performance of DDOR (red) for a dynamically changing swarm of varying network graph degree (green). The round trip time is the measured duration between omnicast counter increments. The simulation run starts with a 2-D configuration, and then switches to a 3-D configuration and repeats the changes in density (also refer to the images in figure 9). The change to 3-D occurs at approximately t=68000.

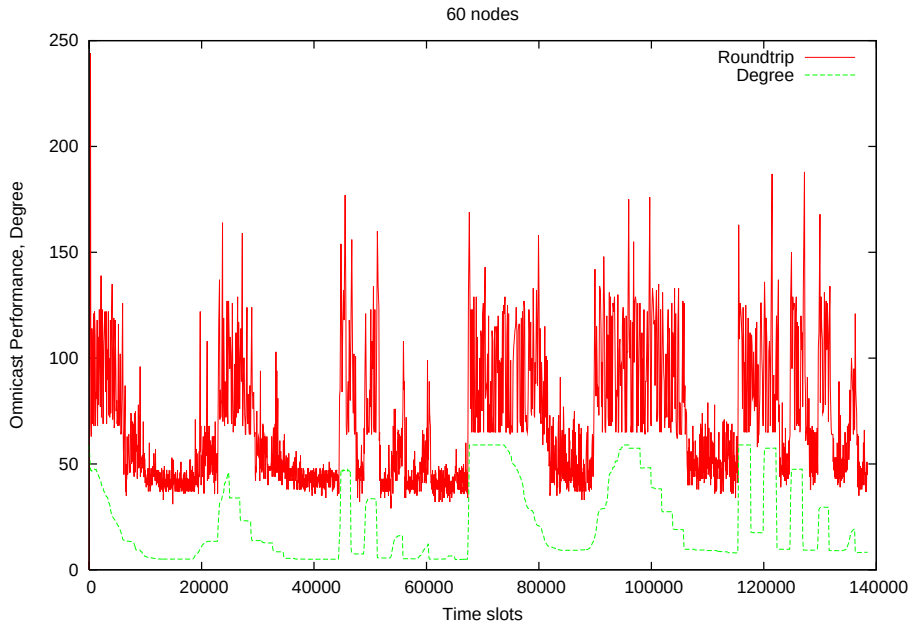


Figure 10. Full simulation run of DDOR with 60 nodes.

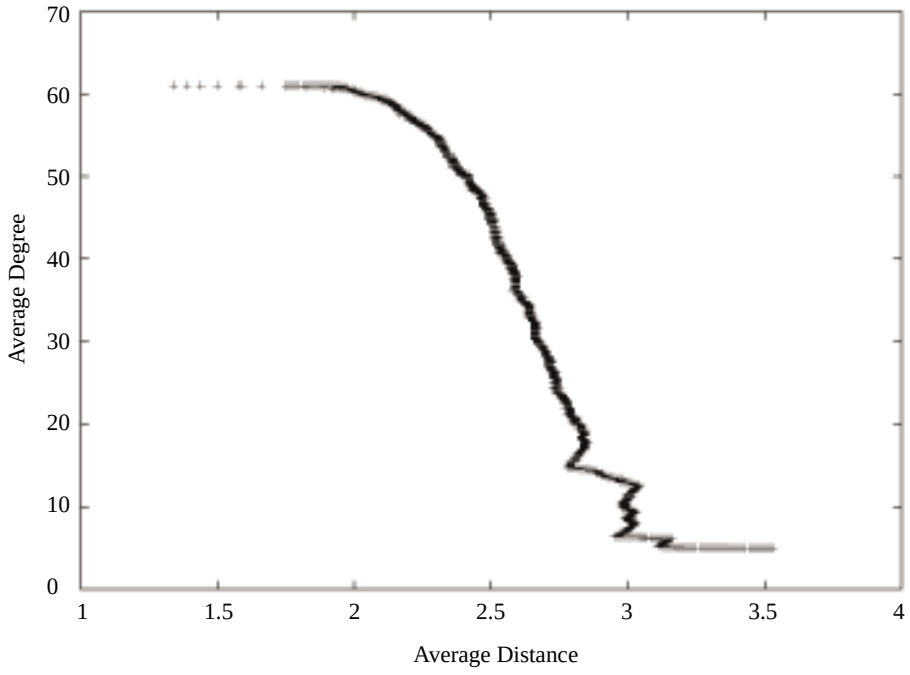


Figure 11. Relationship between average distance and average degree for 62 nodes restricted to a plane

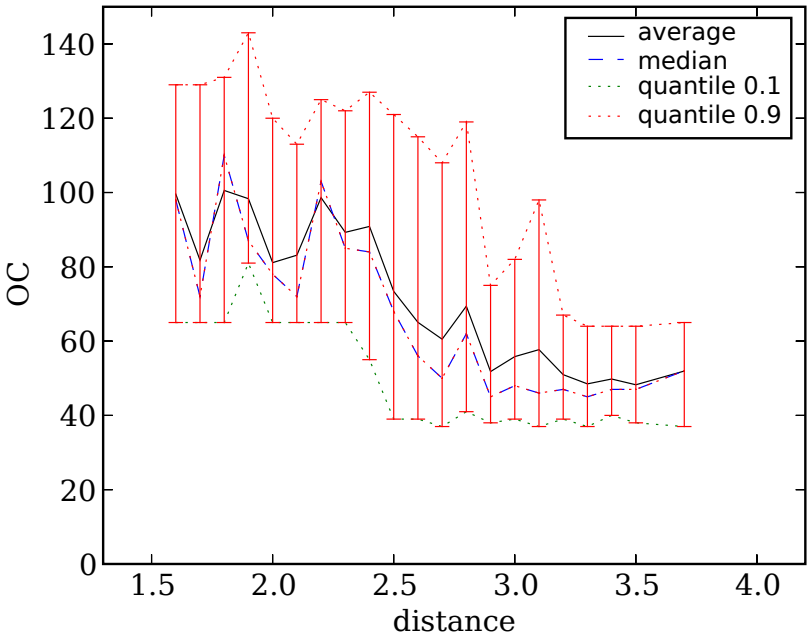


Figure 12. Average distance versus omnicast performance for 60 nodes in a plane

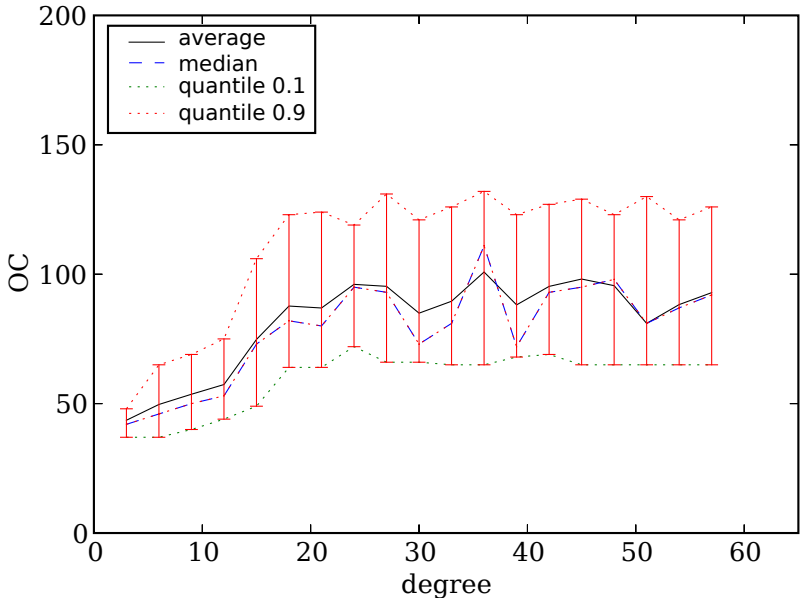


Figure 13. Average degree versus omnicast performance for 60 nodes in 2D and 3D combined

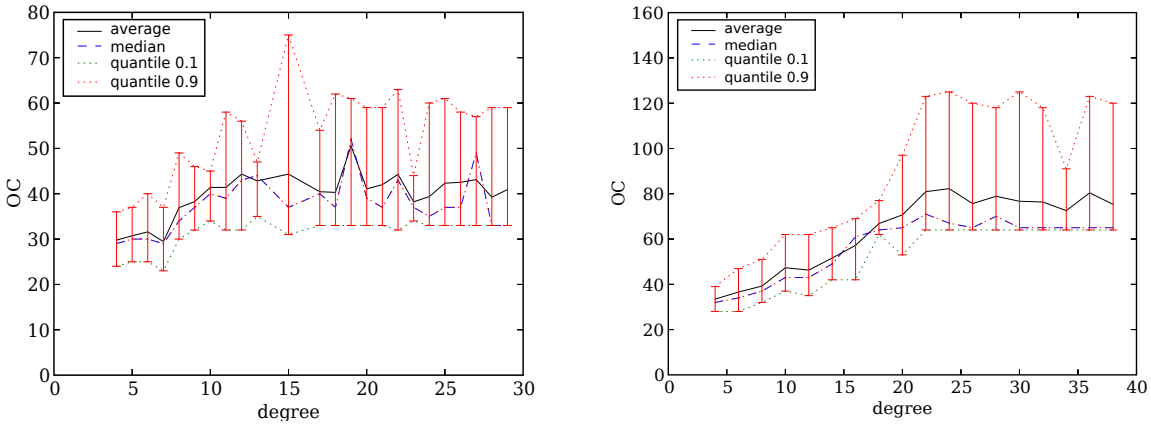


Figure 14. Average degree versus omnicast performance for 30 nodes (left) and 40 nodes (right)

4.4. Results of Simulated Scheduling

As expected the average degree of the network is closely linked to the average distance, as figure 11 shows. The plot in figure 12 shows the omnicast time (the time it takes to perform a full exchange of all local information) over the average distance between submarines. Outliers which are significantly higher than the majority of plot points are due to a reorganisation of the schedule, as the network changes. This usually involves nodes temporarily falling out of the schedule, which results in all other nodes not being able to increment their counter. In a real scenario, submarines would not be affected by lacking updates from only a few units. Units not represented in the schedule are still able to listen to all ongoing information, which means they will not be lost. As can be seen in figure 10, these spikes due to reorganisation are typically lower than 200 time slots for 60 nodes.

Up to a distance of approximately 2.3 meters the performance is quite stable around 100 time steps. This is not surprising, considering that the average degree is still high enough for most nodes having a two hop neighbourhood which contains all other nodes. This means that there is no concurrent communication, and the schedule is basically identical to the schedule of a fully connected network. The reason why the omnicast time is higher than the number of nodes lies in the statistical omission of transmissions as described earlier. Omitted messages slightly delay the global information exchange by up to the duration of one schedule round. For a fully connected network, a schedule round is as many time steps as there are nodes in the network, rounded up to the next power of 2 (in this case 64 time slots).

As the average distance increases, the network connectivity goes down. This allows for more concurrent communication, since 2-hop neighbourhoods do not overlap any more. The omnicast time drops to less than half, with an average of 45 time steps, and maintaining the best performance as the average distance approaches the maximum sensing range. It might appear counter-intuitive that the best performance is reached as the swarm reaches its maximum size, but there is a logical explanation, as indicated before.

Figure 13 shows the same experiment plotted against the average degree of the network. The left side of this plot roughly corresponds to the right-hand side of the previous plot. The best performance is reached for an average degree of 5. This describes the swarm being spread out to almost the maximum communication range, with center nodes having 6 neighbours, and edge nodes having down to 3 neighbours. A low degree allows for concurrent communication in several parts of the swarm at any time. The diameter of the network graph in this configuration is 7. Even though not visible in the plots of the global performance,

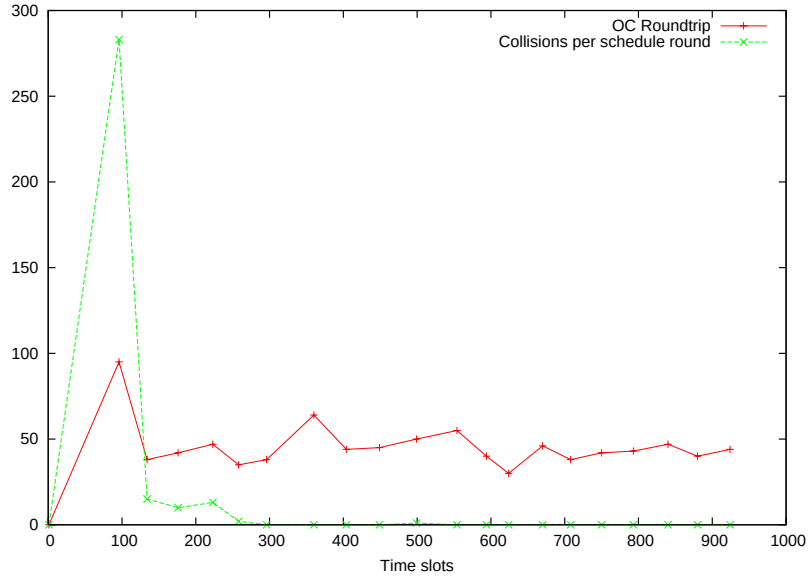


Figure 15. Startup of a network with 45 nodes.

it is obvious that the local schedule lengths is much lower for low degrees (the upper bound is the size of the 2-hop neighbourhood). This results in quicker updates for nodes from their direct neighbours, which is important for swarm control.

Figure 14 shows similar results for networks with 40 respectively 30 nodes. In all cases the maximum performance is reached for the largest-possible expansion of the network without disconnection.

Both theory and simulation results indicate that a low degree is preferable over networks of diameter of 2 or less, mainly because those networks inhibit concurrent communication, and inherently do not scale well. Inversely, this means that for a given geometric swarm size, the communication links should be just long enough to reach the nearest neighbours of each swarm member, but not longer. Short range links are better with regard to local update frequency, global information exchange, and also of course with respect to power consumption.

4.5. Dynamic response

One of the most important aspects of a scheduling algorithm for swarming is the responsiveness to dynamic changes in the network. The communication system must not break down when the swarm changes its configuration, or else the coherence can not be maintained reliably.

The most severe change to the schedule of each node occurs during startup, when no previous schedule exists. The startup is therefore the worst case test scenario for adaptability. Figure 15 shows the behaviour during the first 1000 time slots. During initialisation the number of collisions is high, while the schedules are established. The first omnicast roundtrip (global information exchange) has been completed after less than 100 time slots. After 150 time slots, the performance is already at its optimum and remains stable. The last collisions are resolved after less than 300 time slots. At a typical sending frequency of 10 Hz, the communication system has stabilised after less than 30 seconds.

The dynamic response of the algorithm can be seen in detail in figure 16. The network changes its

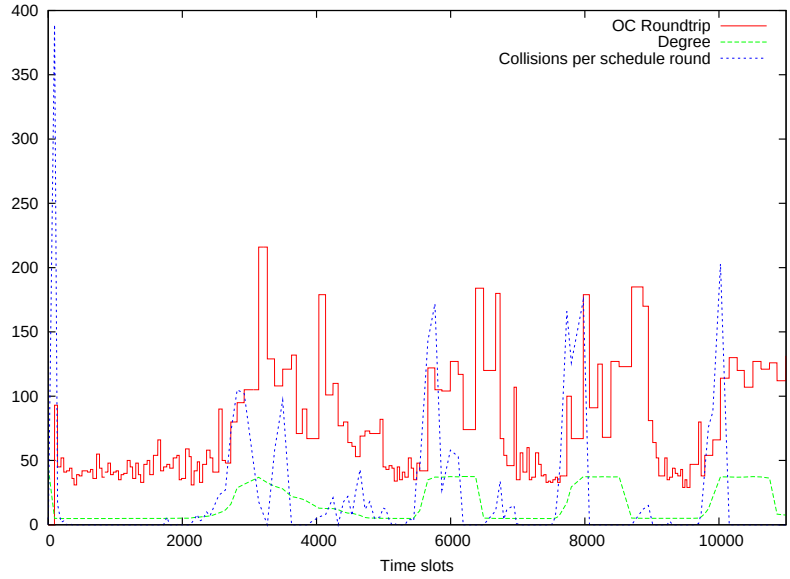


Figure 16. Startup and step response to a change in connectivity for 45 nodes

connectivity first slowly and then rapidly from a network graph degree of approximately 44 (fully connected) to a degree of 6 (fully spread out). The change in degree is shown by the green curve. The red curve shows the measured omnicast roundtrip time, measured by the distributed counter as described earlier. Occuring collisions are counted per schedule round, and plotted in blue. A collision is counted for each node that has two or more transmitting nodes within its range. Counts can quickly rise in dense networks, because many nodes are affected if two nodes transmit at the same time. In networks of low degree, only few nodes are affected, hence the number of collisions is typically lower.

After a short delay the scheduling algorithm detects the change in network topology and reschedules. The rescheduling happens quickly enough so that the effect on communication performance is small. The step from high degree to low degree causes the roundtrip time to double for one schedule round. This is because the schedule for high connectivity (only one node per time slot) is still active, but the lower connectivity inhibits message exchange. Subsequently the schedules get optimised over the next 5-10 schedule rounds, gradually improving performance. The initial delay of about 2-3 schedule rounds is due to the time outs for local removal of disappearing nodes. Nodes are removed from a local schedule if they are not received for more than 2 schedule rounds. During rescheduling, a few collisions may occur if several nodes apply for the same time slot. Collisions are quickly resolved. Once a stable state is reached, there are no further collisions.

The change from low degree to high degree is more easily detected due to the occuring collisions in the denser network. Nodes that were previously transmitting in the same schedule slot are now closer and within each other's range. Once these nodes detect this collision in their received schedules, they immediately apply for a free slot and are quickly rescheduled. The scheduling algorithm adapts to the new topology very rapidly, and reaches a new collision-free stable state.

5. Conclusion

We presented a hardware system that uses GMSK modulation for underwater radio communication. The high spectral efficiency of GMSK has great advantages for communication in salt water. Using electromagnetic waves for underwater communication ensures that links have short range, and long range interference is minimal. This is ideally suited for the presented *Omnicast* scheduling algorithm(DDOR). It could be shown in a simulation that communication networks with a low average node degree allow for faster global information distribution than fully connected networks. This means that short-range links are better-suited for swarm communication. The presented algorithm rapidly responds to dynamic changes in the network topology with only small impact on the performance. The complete system of the GMSK communication link and the DDOR scheduling algorithm is ideally suited for large scale underwater swarms, and will be put into operation and tested in the near future.

6. Acknowledgements

The authors would like to thank Jochen Trumpf, Uwe R. Zimmer and Leif Hanlen for their support.

References

[1] F. Schill, J. Trumpf, U. R. Zimmer, "Towards optimal TDMA scheduling for robotic swarm communication" In Proceedings Towards Autonomous Robotic Systems, 2005.

[2] B. Tavlı, "Broadcast capacity of wireless networks", Communication Letters, Vol. 10(2), pp. 68-69, 2006.

[3] F. Schill, U. R. Zimmer, "Effective communication in schools of submersibles", In Proceedings IEEE OCEANS06, 2006.

[4] S. L. Hakimi, E. F. Schmeichel, "Gossiping in radio networks", Ars Combinatoria, Vol. 35-A, pp 155-160, 1993.

[5] L. E. Kinsler, A. R. Frey, A. B. Coppens, J. V. Sanders, Fundamentals of Acoustics, John Wiley & Sons, 3 edition, 1982.

[6] D. B. Kilfoyle, A. B. Baggeroer, "The state of the art in underwater acoustic telemetry", IEEE Journal of Oceanic Engineering, Vol. 25(1), pp. 4-27, January 2000.

[7] M. Stojanovic, "Recent advances in high-speed underwater acoustic communications". IEEE Journal of Ocean Engineering, Vol. 26(2), pp. 125-136, April 1991.

[8] R.W.P. King, C. W. Harrison, Antennas and Waves, The M.I.T. Press, 1969.

[9] R.W.P. King, G.S. Smith, Antennas in Matter, The M.I.T. Press, 1981.

[10] R. Dunbar, "The performance of a magnetic loop transmitter receiver system submerged in the sea", The Radio & Electronic Engineer, Vol. 42(10), October 1972.

[11] I.S. Bogie, "Conduction and magnetic signaling in the sea. A background review", The Radio & Electronic Engineer, Vol. 42(10), pp. 457-463, October 1972.

[12] R. Somaraju, J. Trumpf. "Frequency, temperature and salinity variation of the permittivity of seawater", IEEE transactions on Antennas and Propagation, Vol. 54(11), pp. 3441-3448, 2006.

[13] K. Mupota and K. Hirade. "Gmsk modulation for digital mobile radio telephony". IEEE Transactions on Communications, Vol. 29(7). pp. 1044-1050, July 1981.

[14] FX589 GMSK modem application notes, Application note, CML Semi- conductor Products, Oval Park - Langford - Maldon - Essex - CM9 6WG - England, 1997.

[15] P.V. Brennan, Phase-Locked Loops, MACMILLAN PRESS LTD, 1996.

[16] P. Horowitz, W. Hill. The Art of Electronics, Cambridge University Press, 1980.

[17] J. Klapper, J.T. Frankle. Phase-Locked loops and Frequency-Feedback Systems, Academic Press Inc., 1972.

[18] CML Microcircuits, Oval Park - Langford - Maldon - Essex - CM9 6WG - England, CMX589A GMSK modem, April 2002.

[19] F. Schill, U. R. Zimmer, "Distributed dynamical omnicast routing", Complex Systems (intl. Journal), Vol. 16(4), pp. 299-316, 2006.