Towards Optimal TDMA Scheduling for Robotic Swarm Communication

Felix Schill[✦], Uwe R. Zimmer[✦], and Jochen Trumpf[✦][⅄]

[✦]Research School of Information Sciences and Engineering
Autonomous Underwater Robotics Research Group
The Australian National University, ACT 0200, Australia
[⅄]National ICT Australia Ltd., Locked Bag 8001, Canberra ACT 2601, Australia[1]

{felix.schill | jochen.trumpf} @anu.edu.au | uwe.zimmer@ieee.org

Initial results are presented on a new TDMA scheduling problem, which tries to minimise the duration of total information exchange throughout a multi-hop wireless network. A new network communication mode **omnicast** is introduced, which implements many-to-many communication, and is similar to a concurrent multiple broadcast from every node to every other node. It can be shown that the lower bound for this problem for arbitrary connected networks with n nodes is n time steps, a general upper bound is $(n^2 - n)$, and $(2n - 3)$ if the graph modelling the network is Hamiltonian. In fact, more recent results (see acknowledgements) show that a better upper bound is $2n - 2$. Simulation results suggest that a tight upper bound is n. Furthermore, it turns out that allowing collisions improves the results, meaning that collision-free solutions are in general suboptimal. A TDMA scheme which optimizes omnicast, will minimise the time span from where new information is released into the network, until every node received it. It also automatically solves the broadcast and convergecast problem for arbitrary senders, and provides consistent response times and bandwidth for realtime operation. The main application lies in robotic swarm communication, where global parameters and environmental information have to be exchanged and updated with minimal and bounded latency. It will be shown how such a TDMA scheme can be applied to a swarm of autonomous miniature submersibles.

Keywords: **Ad hoc networks, TDMA, omnicast, swarm communication**

[1] *National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council*

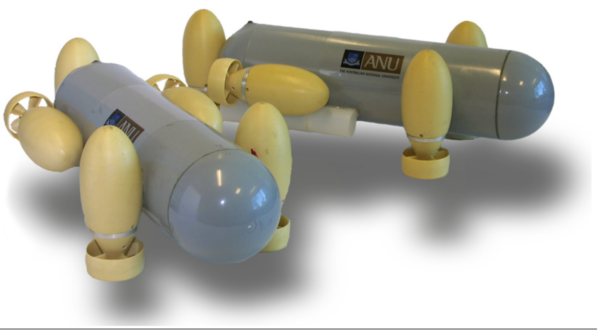
figure 1: Serafina, an autonomous submersible

1. Introduction

In recent years, swarm robotics and sensor networks have attracted great new interest in ad hoc networking, and exposed a variety of theoretical problems, which did not appear in classical networks. These new problems occur on all levels, from channel access, to estimating the network topology, and optimizing information flow. In networks of autonomous, identical nodes, such as robotic swarms and sensor networks, one can observe a paradigm shift: In classical computer networks, it is often assumed that point-to-point communication is the dominating case, and messages are sent from one sender to one dedicated receiver. Furthermore, it is also often assumed that transmissions are sporadic, which favours contention-based access schemes like ALOHA and CSMA.

However, in decentralised swarms or sensor networks, this is no longer the case. Especially in robotic swarms, control parameters have to be exchanged continuously, with short latency and highest possible update rate. Also, in a network of identical nodes, it is counterintuitive to specify a dedicated receiver – messages are rather sent to either the direct neighbourhood (local gossip), or to everyone (broadcast). Since the underlying communication channel is in most cases a broadcast-oriented MIMO channel – this is true for basically all radio networks and bus-oriented networks – it is reasonable to exploit that feature.

There has been a lot of work in recent years on broadcasting in ad hoc radio networks. In [6], the authors present a deterministic broadcasting algorithm with running time $O(n \log n \log D)$ for a network with n nodes and diameter D. Other deterministic algorithms are presented in [3]. A randomised protocol is presented in [1], which achieves broadcast with probability $1 - \varepsilon$ in $O((D + \log n / \varepsilon) \log n)$. The authors point out that there exists an exponential gap between randomised and deterministic protocols. It is interesting to note that the use of randomised protocols does not avoid collisions, which might be an advantage. [4] proves the existence of a schedule for broadcast with $O(D + \log^5 n)$, and specifies a randomised protocol.

In the context of underwater robotics, communication channels are usually of very short range, and also quite slow compared to in-air communication. The autonomous miniature submersible *Serafina* (figure 1) will be equipped with a longwave radio transceiver, which has a maximum transfer rate of 8192 bit/sec, and an optical communication module with 57 kbit/sec transfer rate [8]. The communication range under water is only a few meters. The goal is to achieve an algorithm for a robust, decentralised ad hoc network, which minimises latency, maximises usable bandwidth, and allows to exchange control parameters throughout the swarm in minimal time.

Especially in swarms of robots, communication is not sporadic, but rather a continuous flow of messages and updates, to distribute control parameters, environmental information, and to fuse sensor data. In addition, for all realtime applications, it is crucial to know upper bounds on response time and latency. Time Division Multiple Access (TDMA) protocols have several advantages here over contention-based protocols, which are commonly used in computer networks. One big advantage is that they do not require collision detection, which allows the use of simpler hardware (radio modules or optical transceivers). Another advantage is that they maximise the usage of the available bandwidth. The TDMA scheduling problem has a lot of similarities with the broadcast problem. The papers [5] and [7] present distributed TDMA algorithms, and the latter gives insight into the performance of these algorithms with regard to broadcast, convergecast and local gossip, under the assumption that nodes are arranged in a grid. This is an interesting first step towards optimization of information flow, but does not provide a theoretical foundation for such an analysis. Obviously any given TDMA schedule has a great impact on the information flow, and the speed of information dissemination in the network. Unfortunately the underlying principles are not well understood.

There are basically four modes of communication: one-to-one, one-to-many, many-to-one, and many-to-many. While the first three modes are often and commonly referred to in network theory (unicast, broadcast/multicast, and convergecast), interest in the last mode many-to-many seems to be rather new in the context of communication networks. A very important problem especially in swarm control and formation control is the exchange of certain parameters throughout the entire network. This might be control commands from the operator, which would correspond to a broadcast, but it is more importantly the exchange of parameters from every member of the swarm to all the other members. This includes, but is not limited to, finding a consensus on the direction and speed of the swarm, i.e. to find the slowest member in order not to loose anyone, estimating the swarm density, the center, size or shape of the swarm, or to calculate gradients and extrema on external sensory data such as temperature, brightness, pressure, salinity, just to name a few examples. This requires efficient communication from everyone to everyone. As a continuation of the communication modes 'one-to-one' (unicast), 'one-to-many' (multicast), 'one-to-everyone' (broadcast), and 'everyone-to-one' (convergecast), we would like to propose the name 'omnicast' for 'everyone-to-everyone''. A formal definition follows below.

Omnicast can be applied to TDMA scheduling algorithms. A schedule which solves *omnicast*, can be used as a TDMA schedule, that optimizes overall information flow throughout the network. Furthermore, this approach allows to specify bounds on the time it takes to distribute information throughout the network, which is important in realtime and control applications.

2. 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 $(u, u) \in E$, $\forall u \in V$, even though a node is not assumed to be able to send and receive at the same time. 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, we say a collision occurs at v. In case of a collision, node v can not decode any of the transmitted messages, and can in general not detect the collision, meaning it can not distinguish between a collision and noise. Data integrity can still be guaranteed, since v can use higher-level protocols, such as cyclic redundancy check, to verify messages. Noise and distorted transmissions will then be ignored.

More specifically, it is furthermore assumed for simplicity, that the network graph is symmetrical and connected. It is also assumed that transmissions take place in fixed time slots, and that nodes can synchronise and keep synchronisation with each other – this can be done by monitoring other node's transmissions, and synchronise an internal timer to their messages. To simplify the theoretical analysis, we assume that time is discrete, with one time step as the basic unit, and that the actual transmission of data happens instantaneously within one time step. Practically, this means that time slots have to be long enough to accommodate all occurring message types at the given bandwidth. To implement omnicast, a message has to have space for one datum for each node - this means that the message size depends on the size of the network, and that a bound for the maximum number of nodes has to be known. For some applications, the message size can be constant. Computing a maximum of a distributed measurement requires only the exchange of a single datum in every message. After convergence of an implicit omnicast, every node is guaranteed to have the global maximum value.

3. The omnicast problem

The definition of omnicast has three parts: the start state, the communication phase, and the end state.

Definition 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 t_f is reached, when all nodes have the full set with all tokens, $I_u(t_f) = \{B_v | v \in V\}$ for all $u \in V$.*

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

Definition 2: (The Optimal Omnicast Problem) *Find a schedule $S_G = (T_1, \ldots, T_t)$, $T_i \subset V$ for $i = 1, \ldots, t$, 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.*

It is important to understand, that even though the outcome of an omnicast is equivalent to multiple concurrent broadcasts initiated by each individual node, or also a convergecast to some node followed by a broadcast, the process is quite different. The major difference is that all nodes need to be able to accumulate information, and bundle it in one message. Also, an omnicast can be performed in less time steps than the two alternatives, as will be shown later.

4. Analysis

Let us first find a lower and upper bound for the optimal omnicast. A lower bound shall be defined as a lower bound on the worst-case number of time steps for the best algorithm, working on arbitrary connected graphs. It is not the minimal number of time steps it will at least take for all graphs – but for each algorithm, there is a worst case for which it can not be solved faster than the lower bound. A lower bound is usually specified as a function on properties of the graph, such as the number of nodes n, the diameter D, etc. This definition is in accordance with the literature. Formally, let

$$\mathcal{A} = \{A: \mathcal{G} \rightarrow S \,|\, A \text{ solves omnicast}\} \tag{1}$$

be the class of all algorithms (or functions) A, that solve the omnicast problem and that map from a subclass $\mathcal{G}$ of all connected graphs to the class of all schedules S. Let $|A(G)|$ denote the number of time steps t of that schedule. Then, a function $L: \mathcal{G} \rightarrow \mathbb{N}$ is called *lower bound*, if it fulfils the following condition:

$$\forall A \in \mathcal{A}: \exists G \in \mathcal{G}: |A(G)| \geq L(G) \tag{2}$$

The *absolute lower bound* $L_a: \mathcal{G} \rightarrow \mathbb{N}$ is the minimum number of time steps any algorithm needs for any given graph:

$$\forall A \in \mathcal{A}: \forall G \in \mathcal{G}: |A(G)| \geq L_a(G) \tag{3}$$

An *upper bound* $U: \mathcal{G} \rightarrow \mathbb{N}$ is an upper bound for the worst case number of time steps for the best algorithm, and is defined likewise with this condition

$$\exists A \in \mathcal{A}: \forall G \in \mathcal{G}: |A(G)| \leq U(G) \tag{4}$$

Obviously, since omnicast solves broadcast, it can not be faster than an optimal broadcast. That means that a lower bound for broadcast is also always a lower bound for omnicast. [2]

Theorem 1: *Let G be a connected graph with $n > 1$ nodes. $L(G) = n$ is a lower bound for omnicast on the network modelled by G.*

Proof: Consider the class of all fully connected graphs. To solve omnicast in this class, each node has to send a message with its token of information at least once. If more than one node sends per time step, no node can receive any information, therefore only exactly one node can send per time step. In this case, after n time steps every node transmitted exactly once, the omnicast is solved, and every node has every token of information. It follows that a lower bound for omnicast in general is $L(G) = |V| = n$. ❑

Theorem 2: *Let G be a connected graph with diameter D_g . Then $L_a(G) = D_G$ is an absolute lower bound for omnicast on the network modelled by G .*

Proof: Consider the shortest path in G with maximum length D_G . Obviously, information has to be exchanged from the start to the end of this path. A particular token of information can only proceed by one node per time step on that path. The token from the start node of the path hence needs at least D_G time step to reach the end node. Omnicast can not be solved faster than this on any graph. ❑

Theorem 3: *Let G be a connected graph with n nodes. Then $U(G) = (n^2 - n)$ is an upper bound for omnicast on the network modelled by G .*

Proof: It is sufficient to show existence of an algorithm that solves omnicast in all cases in not more than $(n^2 - n)$ time steps. Consider an optimal schedule, which can be found by exhaustive search. Assume each node's information state I is described by an n -dimensional bit vector, which describes which tokens of information it owns. In the beginning, each node's vector has exactly one bit set, its own bit. In the end state, every node's vector has every bit set. This means that altogether $(n^2 - n)$ bits have to be set by communicating tokens. In every time step, at least one bit will be set in the whole network, otherwise this time step would be redundant, and the schedule would not be optimal. It follows that an optimal schedule has a maximum of $(n^2 - n)$ time steps, which is therefore an upper bound for omnicast. ❑

A better upper bound can be given for arbitrary connected graphs, as John Hallam pointed out (see acknowledgements):

Theorem 4: *Every undirected finite connected graph can be disassembled, one node at a time, without disconnection.*

Proof: A graph either contains cycles, or it does not. In the former case, remove an edge from a cycle. This does not disconnect the graph. Repeat until the graph does not contain any cycles. A graph without cycles is a tree. Removing a leaf from a tree does not disconnect it. Repeat removing leafs from the tree, until it is empty. The order of nodes being removed from the tree can also be applied to the original graph, disassembling it without disconnection. ❑

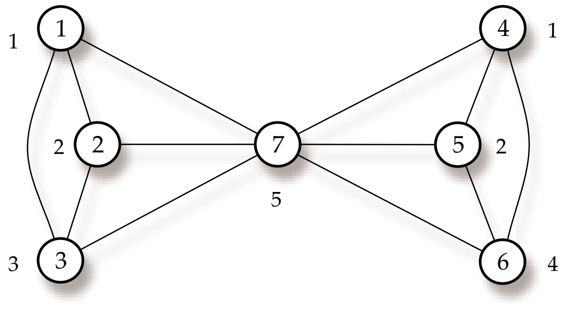

figure 2: A butterfly graph with 7 nodes (numbers next to nodes are transmission time steps)

Theorem 5: *Let G be a connected graph with n nodes. Then $U(G) = (2n - 2)$ is an upper bound for omnicast on the network modelled by G .*

Proof: For induction, assume there is a solution for omnicast with $2(k-1) - 2$ steps, for any connected graph with $k - 1$ nodes. Take a graph G of size k , and remove a node without disconnection. The resulting graph G' has an omnicast solution with at most $2k - 4$. Add two steps to this to obtain a solution for G :

step 1:	the removed node sends.
step $2 \ldots 2k - 3$:	apply the solution for G' .
step $2k - 2$:	any node connected to the removed node sends.

This is an omnicast solution of length $2k - 2$.

Base: The trivial graph of size 1 requires 0 steps. ❑

5. Solutions for special classes of graphs

It has already been shown in theorem 1, that an optimal solution for fully connected graphs requires exactly n time steps. It is obvious that any complete enumeration of all nodes in G corresponds to an optimal solution.

It is possible to extend this to all graphs of radius one. In this case, there exists a node (the center), which is connected to all other nodes. It is always possible to construct a schedule with n time steps, by simply letting all nodes except the center node send one after another, and let the center send as the last node. The center will have accumulated all information by then, and a single transmission from the center reaches all other nodes, completing the task. These solutions are optimal among the solutions without collisions, but are not necessarily optimal among all solutions. This can be illustrated by a counter example.

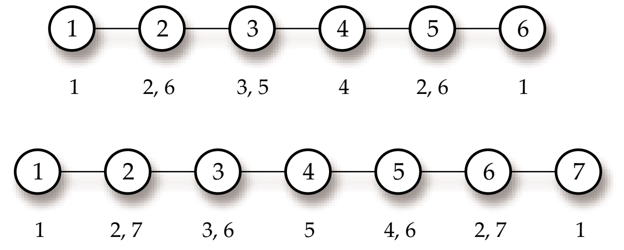

figure 3: Optimal schedules for line graphs. (numbers next to nodes are transmission time steps)

Imagine a graph $G = (V, E)$ with radius 1, for which it is possible to partition its nodes into three disjunct, non-empty subsets $V_l, V_r, C \subset V$, such that $|C| = 1$ and the subgraphs $(V_l \cup C, E)$ and $(V_r \cup C, E)$ are fully connected, and $\neg\exists u \in V_l, v \in V_r | (u, v) \in E$. This class of graphs shall be called *butterfly graphs* (Figure 2 shows an example with 7 nodes). It is now possible to independently and concurrently solve omnicast on the subgraphs (V_l, E) and (V_r, E). Since V_l and V_r are fully connected, this takes $\max\{|V_l|, |V_r|\}$ time steps. The center node in C will have experienced a collision in every time step, and will therefore not have received any information. If we make sure that the last node sending in V_l and respectively V_r may send exclusively, only one extra time step is required. In this scenario, the center node will have received all information from V_l and from V_r, which it now can transmit to all other nodes in V. The overall number of time steps required for networks modelled by butterfly graphs is therefore $\max\{|V_l|, |V_r|\} + 2 \le n$. This shows that omnicast can be solved in less than n time steps for butterfly graphs, if $|V_l| > 1$ and $|V_r| > 1$. On the other hand, if collisions are not permitted, no two nodes can send at the same time, or else the center node would experience a collision. Since every node has to send at least once, a collision free solution for omnicast requires at least n time steps in butterfly graphs. It follows that avoiding collisions yields suboptimal solutions.

Another class of graphs is symmetric graphs of diameter $n-1$, that is, graphs which are a single line (figure 3). We can construct a solution with exactly n time steps for all graphs of this class. Assume an undirected graph

$$\begin{aligned} G &= (V, E) \text{ with} \\ V &= \{v_1, \ldots, v_n\}\text{, and} \\ E &= \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}\ . \end{aligned} \tag{5}$$

A solution for a schedule S_G is of the following form: $S = (T_1, \ldots, T_n)$, with

$$T_i = \begin{cases} \{V_i, V_{n-i+1}\} & ;i = 1\ldots\lfloor n/2\rfloor - 1 \\ \{V_{\lfloor n/2\rfloor}\} & ;i = \lfloor n/2\rfloor \\ \{V_{\lceil n/2\rceil+1}\} & ;i = \lfloor n/2\rfloor + 1 \\ \{V_{\lceil n/2\rceil}\} & ;i = \lfloor n/2\rfloor + 2 \\ \{V_{i-1}, V_{n-i+2}\} & ;i = \lfloor n/2\rfloor + 3\ldots n \end{cases} \tag{6}$$

This applies to graphs with an either odd or even number of nodes. Nevertheless, only for graphs with an even number of nodes, this solution is collision free. For graphs with an odd number of nodes, the center node will experience a collision in time step $\lfloor n/2\rfloor + 3$, and it should be noted that in this case there is no collision-free solution with only n time steps. It can be shown that this solution is optimal for line graphs. The proof exploits the fact that any schedule with $n-1$ time steps (which would be the absolute lower bound) can not solve the problem for line graphs.

6. Simulation results

For analysing optimal schedules on small graphs, an exhaustive search on all schedules was implemented. The runtime of the search algorithm is exponential, so it is only possible to analyse small graphs up to 7 nodes. The search algorithm itself is a hybrid between breadth-first and depth-first. It starts as breadth-first, until the available memory is exhausted, and continues depth-first with limited search depth.

The search algorithm iterates on the time steps of the schedule. For every time step, all possible sets of senders are evaluated. For each possible set of senders, the new information state vector for the network is calculated, and added to the input search space for the next time step. This continues until the end state vector with all bits set is found. In case of the depth first search, the maximum search depth is limited. In the first stage, an optimal collision-free solution is calculated. This converges much faster than a full search, since the number of possible sets of senders is heavily restricted. The length of this solution minus one forms the depth limit for the second stage. In the second stage, a full search is performed. The algorithm starts with a breadth-first search, which runs slightly faster. When it hits the memory limit, it then switches to depth-first search, based on the last output set of the breadth first search. Once a solution is found, it finishes calculation for the current time step, to collect all optimal solutions. During the depth first search, the search depth is being reduced to the number of time steps of the best solution found so far.

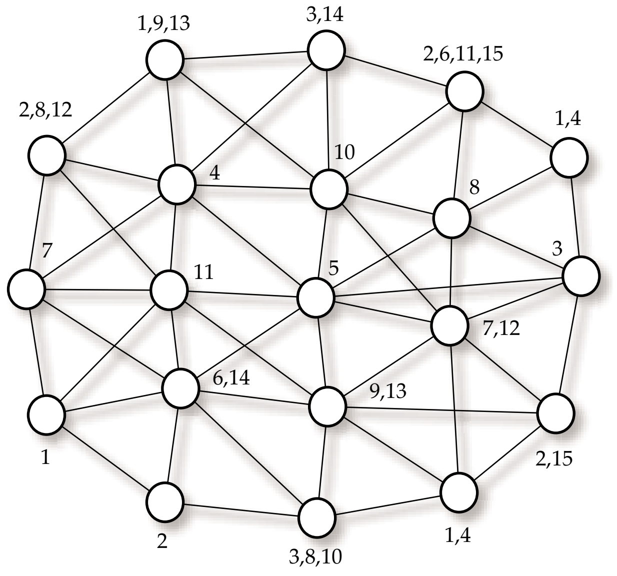

figure 4: A solution with only 15 time steps can be found for this 20-node network.

The simulation program allows to choose between a search on collision-free solutions only, and on all solutions. The set of sets of senders is precalculated, based on a collision analysis on the input graph. Furthermore, a greedy version of the algorithm has been implemented. Here, in every time step, only states with the smallest Hamming norm (the number of bits set) of their information state vector are kept for the next time step.

The described search algorithm is exponential in the number of nodes, and in the number of time steps, which is an unknown function of the number of nodes. Runtimes are therefore extremely long. A full search takes approximately 5 seconds for 5 nodes, 3 minutes for 6 nodes, and 3 days for 7 nodes. The exact times are not relevant for the sake of this article and so the numbers here are only to indicate the order of magnitude for the run-times to be expected on a single-CPU instalment at clock rates of about 2 GHz. Collision-free solutions can be found much faster – approximately 10-15 seconds for 7 and 8 nodes, and 1-2 hours for 9 nodes. Obviously the run time heavily depends on the number of time steps, and therefore also on the complexity of the network. However, the greedy algorithm allows analyses of networks with up to 25 nodes in reasonable time.

The greedy algorithm delivers only sub-optimal solutions. Interestingly, it usually performs better, if the search space is restricted to collision-free schedules.

The most surprising result is, that in most cases, the optimal solution does contain collisions. For a butterfly network with 7 nodes, shown in figure 2, a collision-free solution requires 7 time steps, the optimal solution with 2 collisions only requires 5 time steps. This can be easily explained by the fact, that even though one node suffered a collision, several other nodes did not, and could still successfully decode the message. The information gain outweighs the disadvantage of a collision.

Furthermore, we could not find any graph up to now, for which the optimal omnicast would require more than n time steps.

The 20-node graph with diameter 4 shown in figure 4 has a solution of omnicast with only 15 time steps. This solution has been found with the greedy algorithm, searching only collision-free schedules. There is most likely a better solution, which is also expected to contain collisions. Unfortunately, there is no feasible way to find optimal solutions for a graph of this size. Even though this solution is suboptimal, it further strengthens the hypothesis, that n is a tight upper bound for omnicast. The solution also gives valuable hints how to construct a local heuristic algorithm – it can be observed, that very often neighboured nodes send consecutively. This has the advantage, that a sending node can further distribute information, which it just received. Or, to put it another way, every token of information moves further every time step without delay. A similar argument applies to Hamiltonian graphs, where it has been shown that omnicast can be solved in linear time. This seems to be a useful starting point for a distributed local TDMA scheduling algorithm. If existing TDMA scheduling algorithms can be modified in a way, so that they try to assign successive time slots to neighboured nodes, they should show improved performance with regard to the omnicast problem, and also the speed of information dissemination.

7. Conclusions

A new communication mode *omnicast* has been introduced, as a logical continuation of the *unicast*, *broadcast*, *multicast* and *convergecast* terminology. A theoretical absolute lower bound for omnicast on a network with n nodes and diameter D is $D-1$ time steps. A lower bound is n time steps, and an upper bound is n^2-n steps for arbitrary connected networks. Recent results could improve the upper bound to $2n-2$ steps. For networks which have a Hamiltonian connectivity graph, an upper bound is given by $2n-3$. Constructive optimal solutions for various classes of graphs are presented. Furthermore, it is demonstrated for the class of butterfly graphs, that allowing collisions can significantly reduce the number of required time steps. Simulation results suggest that for most graphs, all optimal solutions contain collisions. It was not possible yet to find a graph for which an optimal solution for omnicast would require more than n time steps – this suggests that n might be a tight upper bound for omnicast.

The results for the omnicast problem can be applied to TDMA scheduling, in order to achieve information dissemination in minimal time. Since in general there will be a whole set of equally optimal omnicast solutions, further second level optimization can be done to maximise the number of concurrent senders, improve local throughput, or minimise energy. Even though it is not feasible to try to calculate an optimal solution for omnicast, the upper and lower bounds provide a frame against which suboptimal solutions can be evaluated.

8. Future work

The most important next step is to apply the results of the global analysis of optimal omnicast to distributed, local TDMA scheduling algorithms. The analysis of Hamiltonian graphs, and the presented optimal solutions for special classes of graphs have in common, that a sender in time step n has a neighbour which has been sending in time step $n-1$. This seems to be a reasonable heuristic, that can be incorporated in existing TDMA protocols. The performance analysis of TDMA schedules with regard to omnicast will be subject to following publications. Further analysis and extensive simulation has to be done to obtain criteria by which it can be locally decided if accepting a collision is beneficial. The final goal is a distributed local algorithm for TDMA scheduling, which comes as close as possible to the performance of globally optimal omnicast. Since this algorithm will then be implemented on miniaturised longwave radio transceivers and optical communication modules for miniature submersibles, computational complexity and memory requirements must be within tight limits. Finally, the dynamical stability of the protocol has to be ensured, to enable swarm communication among mobile robots moving in three-dimensional space. An open theoretical problem is the missing proof for the hypothesis, that a tight upper bound is n steps.

Acknowledgements

We would like to thank John Hallam, University of Southern Denmark, for providing the proof of the linear upper bound with $2n-2$ steps (theorem 5).

References

[1] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. *On the time-complexity of broadcast in multi-hop radio networks: an exponential gap between determinism and randomization*. J. Computer and System Sciences, 45(1):104–126, 1992.

[2] Danilo Bruschi and Massimiliano Del Pinto. *Lower Bounds for the Broadcast Problem in Mobile Radio Networks*. Distributed Computing, 10(3):129–135, 1997.

[3] Bogdan S. Chlebus, Leszek Gasieniec, Anna Östlin, and John Michael Robson. *Deterministic Radio Broadcasting*. In ICALP, pages 717–728, 2000.

[4] Iris Gaber and Yishay Mansour. *Centralized Broadcast in Multihop Radio Networks*. Journal of Algorithms, 46(1):1–20, 2003.

[5] Ted Herman and Sébastien Tixeuil. *A Distributed TDMA Slot Assignment Algorithm for Wireless Sensor Networks*. In ALGOSENSORS, pages 45–58, 2004.

[6] Dariusz R. Kowalski and Andrzej Pelc. *Faster Deterministic Broadcasting in Ad Hoc Radio Networks*. SIAM Journal Discrete Mathematics, 18:332–346, 2004.

[7] Sandeep S. Kulkarni and Umamaheswaran Arumugam. *TDMA Service for Sensor Networks*. In ICDCS Workshops, pages 604–609, 2004.

[8] Felix Schill, Uwe R. Zimmer, and Jochen Trumpf. *Visible spectrum optical communication and distance sensing for underwater applications*. In Proc. ACRA 2004, 2004.