Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online
CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-13 (2016) 259–264

# The method of saving data integrity for decentralized network of group of UAV using quantized gossip algorithms

**Kirill Tyushev** * **Konstantin Amelin** *,** **Boris Andrievsky** *,***

* *St. Petersburg State University, St. Petersburg, Russia (e-mail: kirill.tyushev8@gmail.com, konstantinamelin@gmail.com, boris.andrievsky@gmail.com)*
** *Norwegian University of Science and Technology, Department of Engineering Cybernetics, Trondheim, Norway*
*** *Institute for Problems of Mechanical Engineering of RAS, Russia*

**Abstract:** In this work the method of communication and maintenance of integrity of data in the decentralized network of autonomous mobile robots is suggested. The software and hardware feasibility of such method is proposed. For collective storage and maintain integrity of data in decentralized group of mobile robots the algorithm of checksum calculations is proposed. Quantized gossip algorithms are considered as basis of checksum calculations algorithm. The local voting protocol is used for load balancing of mobile robots. For the programming implementation of the communication of data multi-agent technology is used. An ultra light, unmanned aerial vehicle (UAV) is considered as the mobile robot.

Keywords: Load balancing, UAV, consensus achievement, multi-agent networks, quantized gossip algorithms, mobile robots, data storage.

## 1. INTRODUCTION

The use of various types of mobile robots to perform tasks of monitoring and investigating areas is increasingly becoming a part of our daily lives Wulff et al. [2013], Apvrille et al. [2014], Samad et al. [2013] and etc. In recent years, there has been an increasing interest in groups of mobile robots, rather than isolated machines Amelin et al. [2013a,b], Yoshida et al. [2014], Sukop et al. [2014]. In most cases, during the implementation of group work, a centralized control scheme, in which there is a common leader and ground control center, is used. The main disadvantages of this approach are the need for constant communication of each member with the leader and the need for the leader to work in the group. To solve this problem a decentralized management system can be used.In recent year, the multi-agent systems is widely used as the solution of this task. Where is the main active elements are agents that have the same importance and functionality to the system Granichin et al. [2013], Van Der Hoek and Wooldridge [2008], Olfati-Saber et al. [2007].Management of a group of robots without a single center or common leader can be achieved via organizing a decentralized network by the use of multi-agent technology. This also gives the opportunity to build difference topology relationships between them. To execute decentralized network with varying topology, the task of hardware implementation must be resolved. Regarding this issue, communication technology must be developed. This includes developing hardware, software and algorithmic parts. One of the main tasks for autonomous mobile robots is to research of area. They need to collect, process and send data to other robots or users Bullo et al. [2009], Yu et al. [2010]. Group of mobile robots has a number of advantage compared to single robot. One of this is higher warranty of task execution due to distributed data collection and storage. Komarov et al. [2008], Chilwan et al. [2014], Granichin et al. [2015]. In case of lost part of the robots of group, for example, due to of damage, we can transmit and store all of collected data between all robots of group. This approach requires a lot of robots internal memory and high speed communication channel. In case of lost part of the robots of group, for example, due to of damage, the easiest way to preserve their integrity is to store a copy of all the data on the other robots on each mobile robot. However this will require a large amount of memory on each of the hard drives of the mobile robots. To avoid the storage of redundant data, RAID-like circuits can be used which can compute checksums and then restore data based on those results Plank et al. [1997]. A lot of attention was paid to obtain the corresponding consensus conditions for such systems (see e.g. Ren et al. [2007], Amelin et al. [2012], Amelina et al. [2012], Lewis et al. [2014], Granichin [2015], Amelina and Fradkov [2012]).

RAID (redundant array of independent disks) - an array of multiple storage devices interconnected by high performance communication and perceived external system as a whole. Such an array must maintain the integrity of the data in case of failure of one or more devices, as well as provide high-speed reading or writing data. Each storage

⋆ This work was supported by RFBR (projects 16-07-00890, 15-08-02640). The survey of the recent developments (Sections 4,5) is performed in IPME RAS under sole support of RSF, grant 14-29-00142.

2405-8963 © 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.07.961

device can be regarded as a fully independent entity, being arbitrarily far from the other devices. In this context, RAID can be represented as a multi-agent system, wherein as a single agent serves one storage device. Moreover, the different agents can be located far from each other and connected by slow and unreliable communication channels, and data transmission is possible only at specific times. We call such storage RAID-like.

Example of RAID-like system is Ceph. Ceph is the distributed storage system which can calculate checksums according RAID-like scheme. But it calculates checksums by centralized and synchronous way. Such an approach could lead to poor performance with a large number of storage devices due to a large load on the central device. The new decentralized algorithm of checksums calculation was developed to solve such problems.

Relevance of new approach for checksums calculation consists of decentralized nature for data exchange between mobile robots. Current algorithms in RAID arrays (all of them based on Read-Solomon codes calculations Reed and Solomon [1960]) require one-time data communication from all robots to central computer which process this data to obtain checksums. If at any time we can not transmit data from some robot then in this time we can not obtain checksum. Such difficulties lead to very slow calculations.

In previous work Amelin et al. [2016] we calculate exact data checksums using local voting protocol. In such approach we need to do all calculations in real numbers. Such method requires high-performance hardware on mobile robots. In this work we use only integer numbers since the basis of a new method is quantized gossip algorithms Kashyap et al. [2007]. New approach can not calculate exact data checksums but it do not require high-performance processors on mobile robots.

This paper proposes a scheme of hardware implementation of communication in a decentralized network of mobile robots, based on an example of ultra-light UAV. The implementation of RAID-like schemes for decentralized computing of checksums on mobile robots is considered. It is shown that the checksum takes up less space in memory than all the data from all of the mobile robots, and that through this system you can recover data from all robots, even in the absence of communication or failure of members of the group. The local voting protocol is used for load balancing of mobile robots Amelina and Fradkov [2014] under noise and delay Granichin and Amelina [2015].

## 2. ALGORITHM FOR CHECKSUMS CALCULATION

Consider a multi-agent system of $n$ agents. Let $i, i = 1, ..., n$ - agent number, $t$ - time, $N = 1, ..., n$ - set of agents, $x^i_{t_k}, y^i_{t_k}$ - number on the agent under the number $i$ at time $t_k$. We calculate and store m types of checksums of data on agents. In this case, the minimum number of agents, which preserves the integrity of the data is equal to $(n - m)$.

For simplicity, we assume that the data is transmitted without interference, without delay, and each agent stores only one number. Checksum calculation takes place in specified intervals. Fig. 1 $T$ - time to compute checksums. Upon successful completion of all calculations on the current range in the future is possible to recover data that has been recorded in advance.

At the beginning of the interval is evaluated integral of $f$ on the number on the agent, and the resulting value is stored in a pre-reserved area:

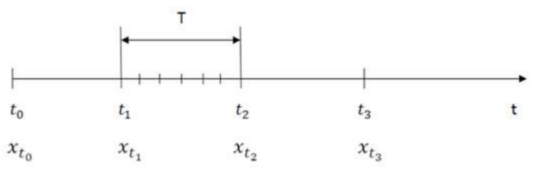


Fig. 1. Schema of checksums calculation

$$y^i_0 = f(x^i_{t_k}), i \in N \tag{1}$$

According to Kashyap et al. [2007] say edge $\{i, j\}$ is selected at time $t$, and let $D^{ij}_t = |y^i_t - y^j_t|$. Then, if $D^{ij}_t = 0$, we leave the values unchanged, $y^k_{t+1} = y^k_t$ for $k = i, j$. If $D^{ij}_t \geq 1$, we require that

$$\begin{cases} y^i_{t+1} + y^j_{t+1} = y^i_t + y^j_t, \\ if\ D^{ij}_t > 1\ then\ D^{ij}_{t+1} < D^{ij}_t,\ and \\ if\ D^{ij}_t = 1\ and\ (without\ loss\ of\ generality)\ y^i_t < y^j_t, \\ then\ y^i_{t+1} = y^j_t\ and\ y^j_{t+1} = y^i_t. \end{cases} \tag{2}$$

In this way, values on agents satisfy the following constraints:

- The value at each node is always an integer.
- The sum of values in the network does not change with time: $\sum_{j \in N} y^j_t = S$ for all $t$.

Let $S$ be written as $NL + R$, where $L$ and $R$ are integers with $0 \leq 1R < N$. Vector $y_t$ converges to quantized consensus distribution $y^i, i \in N$:

$$y^i \in \{L, L+1\}, i \in N, \sum_{j \in N} y^j_t = S \tag{3}$$

Thereby, on each agent we obtain estimated sum of values in the network which is equal $LN$ or $(L+1)N$.

Thus, we have learned to count estimated sum of the numbers $y^j_0$, where $j = 1, \ldots, |N|$.

When we change the function $f$ we change the numbers $y^j_0$, where $j = 1, \ldots, |N|$. This allows to calculate different types checksums for numbers $x^i_{t_k}$, where $i = 1, \ldots, |N|$. Choose functions $f$ and calculate checksums as follows:

$$\begin{cases} f(x^i_{t_k}) = 2^{ij} x^i_{t_k}, i \in N, j \in 1 \ldots m \\ m \leq |N|, x_{t_k} = (x^0_{t_k}, x^1_{t_k}, \ldots, x^n_{t_k}) \end{cases} \tag{4}$$

$$\begin{cases} h_1 = (2^0, 2^1, 2^2, \ldots, 2^{n-1}) \\ \ldots \\ h_m = (2^{m0}, 2^{m1}, 2^{m2}, \ldots, 2^{m(n-1)}) \end{cases} \tag{5}$$

$$\begin{cases} S_1 =< h_1, x_{t_1} > \\ \ldots \\ S_m =< h_m, x_{t_k} > \end{cases} \tag{6}$$

In such approach matrix of linear equations system is Vandermonde matrix whose determinant is not zero. Therefore it is possible to restore the data on $m$ various agents.

After computing the checksum on each agent only one agent keeps it at home, and the rest is removed, so as not to take up too much memory. For simplicity, consider the case of checksum of only one type.

In general, each agent stores an array of numbers. Thus, all numbers can be represented as a matrix whose columns - agents and lines - series of numbers on which the checksums are calculated. On the diagonal of the matrix can be stored initially zeros, and subsequently be stored checksums. Thus, each agent may store the data and checksums:

$$\begin{gathered} x_1^1, x_1^2, 0 \rightarrow x_1^1, x_1^2, S_1 \\ x_2^1, 0, x_2^3 \rightarrow x_1^1, S_2, x_2^3 \\ 0, x_3^2, x_3^3 \rightarrow S_3, x_3^2, x_3^3 \end{gathered} \tag{7}$$

Where the superscript denotes the number of the agent in which it is located, and the subscript - is the number of elements in the array in a given number.

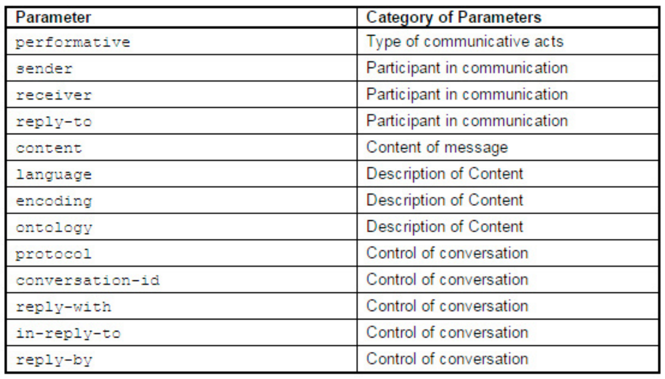
Recover lost files held centrally by solving a system of linear equations in which the unknowns are the lost data. After solving a system of linear equations for each set of numbers in a subscript, it need to convert the files to exact size using the values from the file metadata.

In the simulation allotted agent was used, which had access to all other agents and determined the time of the end of the checksums. In a system of autonomous mobile robots there is no individual member of the group. This would have constant access to all the other mobile robots. Therefore, to determine the end of the checksum calculation, the marker continuously runs between all robots which is used to ensure that the calculated checksum value is the same on all robots. If a robot does not receive the marker for a long period of time, then it generates the marker itself which then circulates in the network.

## 3. FRAMEWORK IMPLEMENTATION

System was developed according to the requirements of decentralized agents interaction. FIPA standards (Foundation for Intelligent Physical Agents) are used as a base of connection between the agents. According to this scheme the agents use Agent Communication Language (ACL). This communication language is very easy to use and quite convenient for realization of checksums calculation algorithm. ACL is defined by the set of parameters included in the message. Base parameter 'performative' defines the language type. Optional parameters are sender / receiver / reply define sender, receiver, and actual receiver of the message. The complete list of parameters is shown in Fig. 2.

Python 2.7 was selected as the language for the framework realization. This language is cross-platform , and python code is short and understandable usually. At this moment latest Python version is 3.5.x, but necessary library PySerial is well written only for version 2.7. Also Python 2.7 is well-validated and has a good feedback within python



| Parameter | Category of Parameters |
|---|---|
| `performative` | Type of communicative acts |
| `sender` | Participant in communication |
| `receiver` | Participant in communication |
| `reply-to` | Participant in communication |
| `content` | Content of message |
| `language` | Description of Content |
| `encoding` | Description of Content |
| `ontology` | Description of Content |
| `protocol` | Control of conversation |
| `conversation-id` | Control of conversation |
| `reply-with` | Control of conversation |
| `in-reply-to` | Control of conversation |
| `reply-by` | Control of conversation |

Fig. 2. FIPA ACL Message Parameters

users. Let's note, that the framework is used in mobile robots with limited computational resources. On the other hand the effect of the language choice on the common required electrical power of the robot may be neglected due to relatively high power of the electrical engines.

Structure of the framework is shown in Fig. 3. Each robot (host) is identified with a single agent. The transmit module transmits messages to other robots and various hardware elements of the robot, and updates the list of robots to send message to at the moment. Algorithm module contains algorithms implemented in the robot. Load balancing algorithm and checksum algorithm are examples of decentralized algorithms that can run on the system.

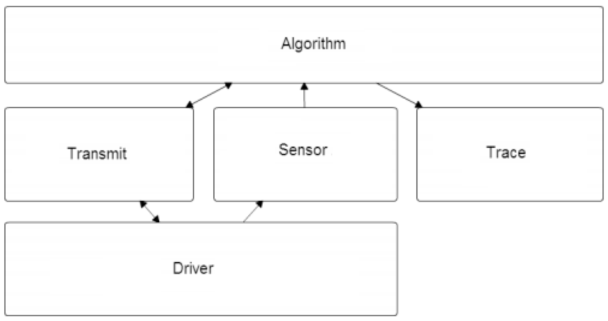


Fig. 3. Framework architecture

A more detailed structure of the system shows the class diagram on Fig. 4.

## 4. THE MESSAGE TRANSFER ALGORITHM

The message transfer algorithm used for sending messages between algorithms on different robots to each other. Module Transmit creates a message queue to each algorithm and put in new messages. The algorithm reads messages from the queue and processes. After reading the message, the algorithm analyzes new data stored in it. When the algorithm has to send any information to other agent, it adds the necessary parameters according to the language ACL in the header and transmits the message to Transmit module with the index of agent this message for. Transmit module adds own header to the message, which specifies the name of the algorithm sent the message, and the number of iteration of the algorithm and transmits to Driver

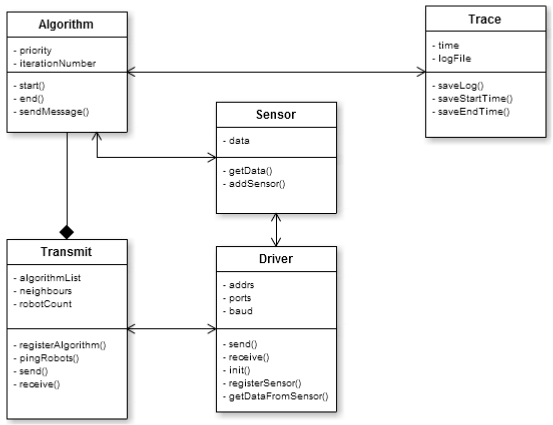


Fig. 4. UML-Diagram of the framework

module. Driver module adds own header, which abstract index of robot-receiver agent is replaced by the MAC-address of that robot. Having formed a packet, module sends it using the physical module data. On the other agents algorithm that targets this message receives and processes it in reverse order.

The transmit module keeps the number of online agents. Transmit module after the request calls function pingRobots() sending the request of connection confirmation to each robot. After receiving a request on the other agent the same Transmit module responds to him out of the lineup. Then, after waiting a certain time interval, the module counts the number of Transmit received confirmations and updates its neighbor table, i.e. robots are connected. If the message from a robot does not come, or comes after the timeout, the connection with the robot is considered to be lost until the next call pingRobots(). A block diagram of a procedure for processing incoming messages Transmit module is shown in Fig. 5.

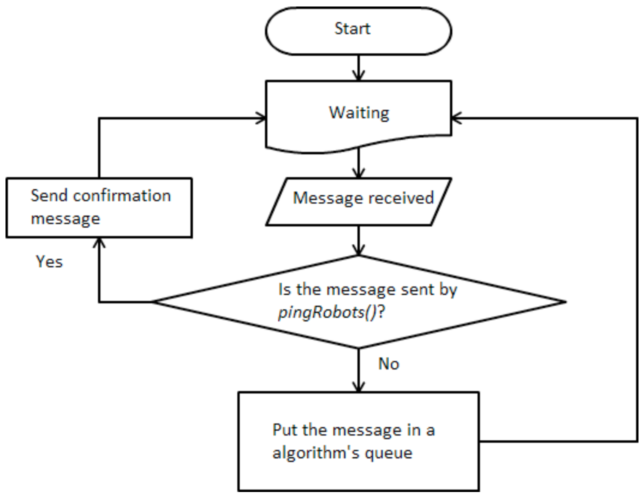

Start
Waiting
Message received
Send confirmation message
Yes
Is the message sent by pingRobots()?
No
Put the message in a algorithm's queue

Fig. 5. Received message processing by Transmit module

Module Trace is used for monitoring the operating time of algorithms processing and to store logs. It counts the time from the start of work of the system. If the algorithm is requested to access some data from the sensors, for example, to get a picture with the camera, the data is provided by module Sensor. For example, consider the message transmission robot algorithm k on the robot n to the algorithm k on the robot m, as shown in Fig. 6: Algorithm module sends a text message to the Transmit module ACL-header, which specifies the code of algorithm-sender, the code of algorithm- receiver, and all other required fields.

Module Transmit, in turn, adds MTP-header, which indicates the address of the robot-sender, the address of the robot-receiver, the iteration number of the algorithm and the other according to the format. Then the module transmits the message to module Driver, which sends a packet on the network. Driver on the robot receives m package, transfer it to Transmit module and Transmit module transfers it to the intended algorithm.

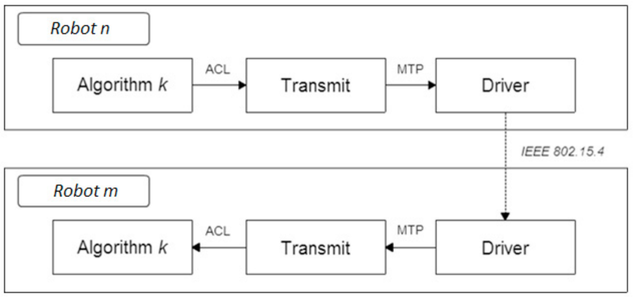


Fig. 6. Example of message sending between algorithms

## 5. HARDWARE

It is proposed to use a three-layer Amelin [2012] management system of mobile robots, the concept of which is the presence on an additional microcomputer (see Fig. 7). Whereby it becomes possible to use multi-agent technology. RaspberryPi model b+, is used as a microcomputer, which is built based on on a processor Broadcom SoC ARM11. By default, OS Debian is installed on the microcomputer.

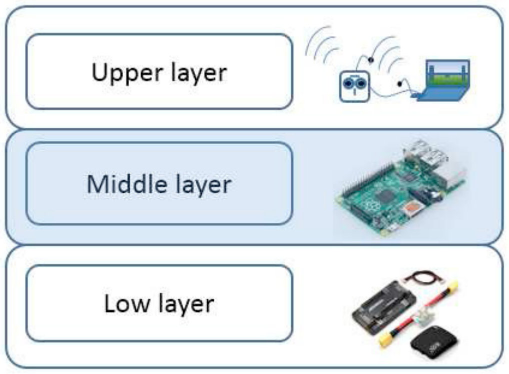


Fig. 7. Three-layer management system

To ensure data exchange, specialized communication modules XBee Series Pro are used. These modules can transmit data within a 1km range, realize radio communication in the group with the absence of a single centre and each device can act as an initiator of the data transmission and each can receive data.

The module operates at a frequency of 2.4 GHz. Transmission rate is 250 bits per second. Devices use a UART interface, which is based on RS-232. Devices can connect

RaspberryPi via the standard means of OS Debian. The modules have a PAN ID (Personnel Area Network ID), which allows them to communicate on the same network without being tied to the addresses of one another. To protect the network XBee modules have the ability to customize the encoding of information transmitted.

## 6. TESTING

For testing and performance measurements, the following experiments were carried out. Several drones were launched. They were photographing the area and in parallel were calculating two types of checksums for the photographs. Upon return to the base, two of the eight UAV were switched off and the others were connected to the base station. While copying files to the base station, missing photos were calculated by checksums and thus all data have obtained.

According to RAID-scheme each mobile robot stores data files and checksum files. All data are divided to stripes. In stripe there is one file from each robot and for each stripe we store duplicate copy of metadata file on each robot. Consider in detail the decentralized calculation of checksums in single stripe (several stripes are computed sequentially):

- Calculate the exact size of the largest file in the stripe - X.
- Each robot loads file into memory, and if file size is less than X, then we supplement it with zeros to size X. If some robot is brought and its file is missing, such file is replaced by zeros of size X.
- On the each robot we divide file into individual numbers and corresponding function for each number is computed depending on the type of checksum.
- For each set of numbers we calculate checksums of desired types.
- Save the resulting checksums on the desired mobile robots.
- Write the exact size of all files in the stripe to metadata file on each mobile robot (After restoring files we need to cut them on source sizes).

Fig. 8 shows dependence of the amount of redundant data (in percent on one UAV) of the number of UAVs, which could be brought down without losing data integrity for 10 UAVs. The checksums on the one UAV take up very less space in memory than all the data from all of the mobile robots.

Fig. 9 shows time of checksums calculation for the case in which the each UAV stores at one photo of 50KB. Robots were organized to the topologies ring and line. Slow performance of checksums calculation caused by low data rate of communication modules XBee that were available. But computations became faster compared with the previous work Amelin et al. [2016] duo to absence of calculations with real numbers.

In this scheme the centralized UAV is absent (all UAVs have equal importance) and if connection between some UAVs is broken for some time then checksums will still be calculated. The UAVs communicated with each other at random moments and thereby they made data exchange at unpredictable moments. But novel algorithm of checksum calculation can calculate checksums in such difficult conditions in contrast to classical algorithms for RAID-arrays. In this way new solution is a good candidate to deal with high data integrity in a group of mobile robots.

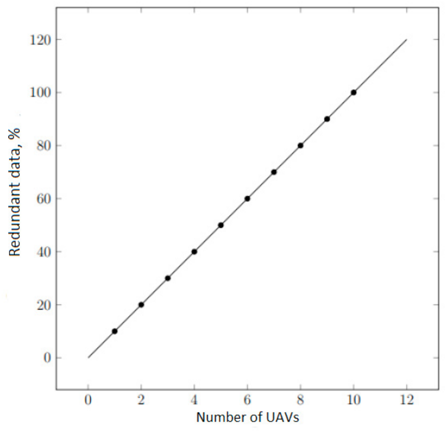


Fig. 8. Amount of redundant data in percent on one UAV

| Robots count, N | Topology "Ring", min | Topology "line", min |
|---|---|---|
| 4 | 3 | 8 |
| 5 | 6 | 20 |
| 6 | 11 | 40 |
| 7 | 17 | 60 |
| 8 | 25 | 82 |

Fig. 9. Time of calculating checksums in minutes

## 7. CONCLUSION

The paper presents method of maintain integrity data in decentralized group of mobile robots, the technology of communication, hardware and software feasibility of using this method. The implementation of a decentralized algorithm for the checksum data with using quantized gossip algorithms is proposed, in order to preserve data integrity when there is a lack of communication with the members, or some of the mobile robots fail. The proposed methods, hardware and software implications of their implementation are approbated on a group of ultra-light UAV VTOL (vertical take-off and landing).

## REFERENCES

K. Amelin, N. Amelina, O. Granichin, and O. Granichina. Multi-agent stochastic systems with switched topology and noise. In *Proc. 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD)*, pages 438–443, Kyoto, Japan, 2012.

Konstantin Amelin, Natalia Amelina, Oleg Granichin, Olga Granichina, and Boris Andrievsky. Randomized algorithm for uavs group flight optimization. *11th IFAC International Workshop on Adaptation and Learning in Control and Signal Processing*, pages 205–208, 2013a.

Konstantin Amelin, Natalia Amelina, Oleg Granichin, and Victor V Putov. Task allocation algorithm for the cooperating group of light autonomous unmanned aerial vehicles. In *IFAC Proceedings Volumes (IFAC-PapersOnline) 2 (PART 1)*, pages 152 – 155, 2013b.

Konstantin Amelin, Kirill Tyushev, and Vasiliy Kaliteevskii. Communication and maintaining of data integrity method for decentralized network of autonomous group of mobile robots. In *Proc. IEEE 2016 3rd International Conference on Control, Decision and Information Technologies (CoDIT-2016)*, Malta, April 2016.

K.S. Amelin. Randomization in control for a small uav fly optimization under unknown arbitrary wind disturbances. *Cybernetics and Physics*, 1(2):79–88, 2012.

N. Amelina and A. Fradkov. Approximate consensus in the dynamic stochastic network with incomplete information and measurement delays. *Automation and Remote Control*, 73(11):1765–1783, 2012.

N. Amelina, A. Fradkov, and K. Amelin. Approximate consensus in multi-agent stochastic systems with switched topology and noise. In *Proc. IEEE 2012 Multiconference on Systems and Control (MSC-2012)*, pages 445–450, Dubrovnik, Croatia, 2012.

Natalia Amelina and Alexander Fradkov. Approximate consensus in multi-agent nonlinear stochastic systems. In *Control Conference (ECC), 2014 European*, pages 2833–2838. IEEE, 2014.

Ludovic Apvrille, Tullio Tanzi, and Jean-Luc Dugelay. Autonomous drones for assisting rescue services within the context of natural disasters. In *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, pages 1–4. IEEE, 2014.

F. Bullo, J. Cortés, and S. Martinez. *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms.* Princeton University Press, 2009.

A. Chilwan, N. Amelina, Z. Mao, Y. Jiang, and D.J. Vergados. Consensus based report-back protocol for improving the network lifetime in underwater sensor networks. *Lecture Notes in Computer Science (LNCS)*, 8846:26–37, 2014.

O. Granichin, P. Skobelev, A. Lada, I. Mayorov, and A. Tsarev. Cargo transportation models analysis using multi-agent adaptive real-time truck scheduling system. In *Proc. of the 5th International Conference on Agents and Artificial Intelligence. (ICAART?2013)*, volume 2, pages 244–249, Barcelona, Spain, 2013.

Oleg Granichin. Stochastic approximation search algorithms with randomization at the input. *Automation and Remote Control*, 76:762–775, 2015.

Oleg Granichin and Natalia Amelina. Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances. *IEEE Transactions on Automatic Control*, 60(5), 2015.

Oleg Granichin, Zeev (Vladimir) Volkovich, and Dvora Toledano-Kitai. *Randomized Algorithms in Automatic Control and Data Mining.* Springer, 2015.

Akshay Kashyap, Tamer Basar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

S.N. Komarov, A.N. Terekhov, and O.A. Granichina. Integrated-distributed automated information system for a large scientific and educational institutions. *Vestnik St. Petersburg University. Ser. 10: Applied Mathematics, Informatics and Control Processes*, (1):87–94, 2008.

F.L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das. *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches (Communications and Control Engineering).* Springer, 2014.

R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

James S Plank et al. A tutorial on reed-solomon coding for fault-tolerance in raid-like systems. *Softw., Pract. Exper.*, 27(9):995–1012, 1997.

I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

W. Ren, R.W. Beard, and E.M. Atkins. Information consensus in multivehicle cooperative control. *Control Systems, IEEE*, 27(2):71–82, 2007.

Abd Manan Samad, Nazrin Kamarulzaman, Muhammad Asyraf Hamdani, Thuaibatul Aslamiah Mastor, and Khairil Afendy Hashim. The potential of unmanned aerial vehicle (uav) for civilian and mapping application. In *System Engineering and Technology (ICSET), 2013 IEEE 3rd International Conference on*, pages 313–318. IEEE, 2013.

Marek Sukop, Mikulas Hajduk, and Rudolf Janos. Strategic behavior of the group of mobile robots for robosoccer (category mirosot). In *Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on*, pages 1–5. IEEE, 2014.

Wiebe Van Der Hoek and Michael Wooldridge. Multi-agent systems. *Foundations of Artificial Intelligence*, 3: 887–928, 2008.

Thorben Wulff, Sascha Lehmenhecker, Eduard Bauerfeind, Ulrich Hoge, Kimberly Shurn, and Michael Klages. Biogeochemical research with an autonomous underwater vehicle: Payload structure and arctic operations. In *OCEANS-Bergen, 2013 MTS/IEEE*, pages 1–10. IEEE, 2013.

Kazuya Yoshida, Hiroaki Fukushima, Kazuyuki Kon, and Fumitoshi Matsuno. Control of a group of mobile robots based on formation abstraction and decentralized locational optimization. *Robotics, IEEE Transactions on*, 30(3):550–565, 2014.

Wenwu Yu, Guanrong Chen, and Ming Cao. Distributed leader–follower flocking control for multi-agent dynamical systems with time-varying velocities. *Systems & Control Letters*, 59(9):543–552, 2010.