ANALYSIS OF THE RESIDUAL INFORMATION POTENTIAL OF A SMART DUST SWARM ACROSS THE LIFE CYCLE BASED ON DATA EXCHANGE AND POST-PROCESSING ALGORITHMS

Igor Kovalev (1,2), Vasily Losev (2), Mikhail Saramud (1,2)*

(1) Siberian Federal University, Krasnoyarsk; (2) Reshetnev Siberian State University of Science and Technology, Krasnoyarsk,
Russian Federation

* Corresponding Author, e-mail: kovalev.fsu@mail.ru

Abstract: The paper is devoted to increasing the quantity and reliability of data collected by a degrading swarm of smart dust particles. Algorithms are proposed that apply to both the particles themselves and to mainframes for post-processing data obtained from particles after their life cycle ends. Scenarios of particle behavior in various data exchange modes are described. A software simulation of swarm behavior in a modeling environment, incorporating data accumulation and particle failures, was conducted. Data confirming the effectiveness of the proposed algorithms were obtained. Behavior scenarios for particles during data exchange are proposed, depending on their information visibility and throughput. Algorithms for tagging potentially unreliable data and an algorithm for disabling the active role of a particle in the case of repeated transmission of such data are proposed.

Keywords: machine-to-machine interaction, smart particles, control modes, communication technologies, information visibility.

1. INTRODUCTION

At present, the concept of the “smart dust” technology stack as an integrated measurement complex of a distributed network of microscale sensor nodes is being actively developed in the direction of the Internet of Things and cyber-physical systems [1]. Miniature autonomous nodes, combined into a swarm, enable the formation of spatiotemporal clouds of environmental parameters, thereby opening new opportunities for monitoring the condition of industrial facilities, mining operations, and complex technogenic systems.

At the same time, such networks inherit the limitations of classical Wireless Sensor Networks (WSNs): limited computational resources and node energy supplies, unstable communication channels, limited information visibility, and high sensitivity to failures of individual elements [2].

A key problem for the practical application of smart dust swarms is ensuring the reliability and integrity of the collected data amid potential hardware failures, malicious interference, and random measurement errors. Scientific publications present a wide range of approaches to data authentication and validation in WSNs, including reversible digital watermarking methods for monitoring the integrity of network packets, intelligent validation algorithms using artificial intelligence methods for detecting multiple measurement errors, as well as data aggregation schemes that take into account the spatial correlation of readings and the uneven distribution of node energy resources [3-5]. However, most existing solutions are oriented toward stationary or low-mobility networks, whereas measurement swarms of smart dust are characterized by dynamic topologies, limited particle lifecycles, and massive simultaneous failures.

A separate research direction concerns voting-based decision-making algorithms for improving resilience to faulty nodes [6, 7]. A scientific study on secure localization in WSNs proposes voting schemes based on clustering and weighted centers of mass, enabling simultaneous estimation of the target location and identification of malicious nodes through statistical analysis of measurements [8].

These approaches demonstrate that the use of metadata, for example, node trust estimates, error distributions, and network cluster structure, increases algorithm robustness compared with classical voting methods [8]. However, integrating such schemes into smart dust swarm architectures requires consideration of the specific features of the particle life cycle, the heterogeneity of their hardware and software platforms, and the limited length of stored data records.

In the context of energy efficiency and system durability, routing and data collection strategies for mobile and static smart dust configurations are being actively investigated. Studies devoted to the search for energy-efficient data routing mechanisms show that optimization of routes and node roles can significantly increase network lifetime without substantial degradation in measurement quality [9, 10]. At the same time, the problem of ensuring energy efficiency, high storage redundancy, and resilience to particle failures simultaneously remains open, especially under conditions of massive node failure in the late stages of the swarm life cycle.

In this regard, an important scientific and practical task is the development of decision-making and data validation algorithms for smart dust swarms and WSNs that take into account the swarm life cycle, the diversity of particle hardware and software platforms, and the use of metadata on node reliability.

This paper proposes a set of algorithms implemented both at the level of individual particles and at the mainframe level, enabling self-diagnostics and detection of particles with hardware failures based on temporal and metrological inconsistencies; dynamic redistribution of transmitted data volume among neighboring nodes to increase the probability of data preservation; and post-processing using modified voting schemes based on metadata and the degree of particle diversity. This enables the swarm to increase its residual information potential in the final stages of its life cycle and to improve the reliability of the resulting data when integrated into information security and monitoring systems.

2. MATERIALS AND METHODS

The mathematical description of the proposed data exchange algorithm in a particle swarm includes formalizing the data structure, defining exchange scenarios based on visibility and bandwidth, and providing an algorithm for bidirectional data transmission between swarm particles. This approach enables the implementation and analysis of swarm behavior under various information-environment conditions.

2.1. Basic Notation

P denotes the set of particles in the swarm.

$p_i \in P$ denotes a source particle.

$R(p_i)$ denotes the set of receiver particles that are "visible" to particle p_i.

$V(p_i) = |R(p_i)|$ denotes the information visibility indicator (the number of visible receiver particles).

$C(p_i) \in |0,1|$ denotes the channel bandwidth of particle p_i (normalized to 1).

$V(p_i) = k \cdot C(p_i)$, where k is the proportionality coefficient (selected according to the problem conditions).

$D_{curr} = (t, n, id_{scr}, id_{dst}, s_1, s_2, s_3)$ denotes the current data set (record).

$D_{arch} = \{(t_j, n_j, id_{scr}, id_{dst}, s_{1j}, s_{2j}, s_{3j})\}_{j=1}^{m}$ denotes the archive data stack (a set of previous data sets).

2.2. Scenario Probabilities

Let $C = C(p_i)$.

If $C \in [0, c_1)$, then scenario 3 is applied.

If $C \in [c_1, c_2)$, then scenario 2 is applied.

If $C \in [c_2, 1]$, then scenario 1 is applied.

where c_1, c_2 are thresholds determined by the problem conditions (for example, proportionally to *V*).

2.3. Data Exchange Algorithm

The graph-analytical representation of the data exchange algorithm in a particle swarm is shown in Figure 1.

Input: p_i, $C(p_i)$, D_{curr}, archive stack D_{arch}

Output: updated data for all exchange participants

1. Calculate $V(p_i) = k \cdot C(p_i)$.
2. Determine the scenario according to the value of $V(p_i)$: if $V(p_i) = 6$, scenario 1 is applied; if $V(p_i) = 3$, scenario 2 is applied; if $V(p_i) = 1$, scenario 3 is applied.
3. For each scenario:
   For each $p_j \in R(p_i)$:
   Transmit D_{curr} from p_i to p_j and back.
   If scenario 2 is applied, also transmit D_{arch1} in both directions.
   If scenario 3 is applied, also transmit D_{curr} and $D_{arch(1\ldots5)}$ in both directions.
4. Update the data stacks for all participants.

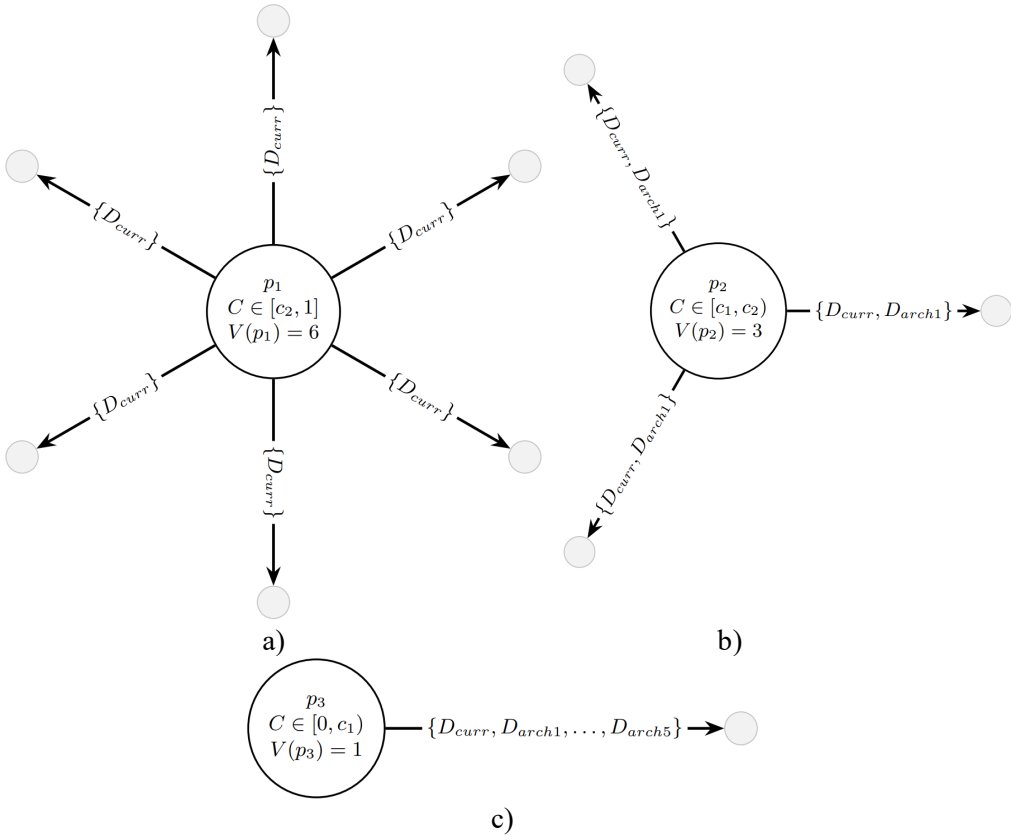


Figure 1. Graph-analytical representation of the data exchange algorithm in a particle swarm: (a) scenario 1; (b) scenario 2; (c) scenario 3

2.4. Formalization of Data Transmission

For each pair (p_i, p_j):

$$\text{Exchange}(p_i, p_j) = \begin{cases} \{D_{curr}\}, & if\ scenario\ 1\ is\ applied \\ \{D_{curr}, D_{arch1}\}, & if\ scenario\ 2\ is\ applied \\ \{D_{curr}, D_{arch1}, D_{arch2}, D_{arch3}, D_{arch4}, D_{arch5}\}, & if\ scenario\ 3\ is\ applied \end{cases} \quad (1)$$

The data exchange algorithm in the particle swarm is formalized as the function:

$$Handshake(p_i) = \begin{cases} \bigcup p_j \in R(p_i)\ \text{Exchange}_1(p_i, p_j), & V(p_i) = 6 \\ \bigcup p_j \in R(p_i)\ \text{Exchange}_2(p_i, p_j), & V(p_i) = 3 \\ \text{Exchange}_3(p_i, p_j), & V(p_i) = 1 \end{cases} \quad (2)$$

where Exchange_k denotes the set of transmitted data sets for scenario *k*.

2.5. Behavioral Characteristics of the Swarm

Achieving the target objective, namely preserving the maximum possible amount of data from other swarm participants while excluding agents with hardware failures from the swarm and performing resource balancing in accordance with agent operating modes, can be carried out at various stages of the life cycle of a smart dust particle swarm.

The algorithms can be divided into several groups (classes):

1. Algorithms applied directly within a particle during its active phase, while collecting and transmitting information;
2. Algorithms applied on the mainframe for processing the entire volume of data obtained from particles after completion of their life cycle.

The algorithms of the first group (class) can be additionally divided into:

1.1 Decision-making algorithms without external control actions;
1.2 Decision-making algorithms taking into account external control actions.

The proposed algorithm for modifying particle operations, aimed at increasing data collection efficiency, adjusts the number of data transmission sessions with neighboring particles based on the number of visible particles. Let us assume that during one iteration, the communication channel bandwidth is sufficient to establish a “handshake” between particles and to transmit and receive six data sets (1), (2). A data set is defined as a record containing the following parameters: timestamp, current iteration number, source particle ID, storage particle ID, and sensor values. If a particle detects six neighbors, then the most reasonable approach is to exchange one data set with each of the six neighboring particles. This increases the probability of data preservation if, by the end of the swarm life cycle, at least one of these seven particles remains operational. If a particle “sees” only three neighboring particles, then it is reasonable to exchange two data sets with each of them; if it sees two neighboring particles, then three data sets should be exchanged. If only one neighboring particle is visible, then six data sets should be exchanged.

After data exchange between particles, a particle may immediately perform certain calculations and manipulations for which the controller's computational power is sufficient. For example, it may compare the timestamp in its own data record for the current iteration with those received from neighboring particles. If the timestamp in one of the data sets for a given iteration differs significantly, by seconds or minutes instead of units or tens of milliseconds, then it can be concluded that a failure has occurred in the controller components of that particle, for example, in the clock generator. Such a failure may lead not only to distortion of system time but also to data corruption.

A comparison of measured parameter values, such as temperature and pressure, for the current iteration is also provided. Since communication sessions are technically possible only between particles that are close together, the parameter values should not differ significantly. All values of the same parameter are compared to determine whether any differences exceed a specified threshold. Based on this analysis, the data may receive a special mark indicating potential incorrectness.

Based on data marked as “incorrect” or “suspicious,” control actions may be applied both within a particle and to other particles. If the same particle transmits incorrect data during 10 iterations, its ID is marked as an unreliable source. All data subsequently

received from the marked particle will also be labeled as suspicious, even if they pass all checks, since it is impossible to predict during which iterations the failure will recur. In addition, a message may be sent to that particle indicating that it has been recognized as faulty. If a particle receives such a message from five or more neighbors, it changes its operating mode. Subsequently, it no longer transmits its own data to other particles, but only receives and stores the maximum possible amount of data from neighboring particles. However, the storage ID is still marked as suspicious, since a particle failure may lead not only to corruption of its own data but also to corruption of correctly received data from neighboring particles. These additional labels occupy only a few bits in each data set; however, they enable improved data reliability during subsequent analysis.

The algorithms applied at the final stage, which significantly influence the quality of the resulting data, are executed on the mainframe to process the entire volume of data obtained from particles after completion of their life cycles. During data post-processing, we are no longer limited by the particle's scarce computational resources or strict timing constraints. At this stage, all surviving data read from particles that remained operational during the final stage of the swarm life cycle, namely information transmission after particle deposition, are available. At this stage, classical voting methods, such as consensus majority voting, may be applied. However, modifications incorporating labels can be proposed to improve data reliability. In addition to the values themselves, we possess information about the ID of the particle that obtained the data from its sensors, the storage ID from which the data set was read, where these IDs do not necessarily coincide, bits marking suspicious or incorrect data, as well as information associated with a particle having a specific ID. Information about the properties of a particular particle may not necessarily be stored in the data table of that particle itself; however, additional information about particles is available from the entire swarm.

In this paper, an example involving 900 particles with three hardware platforms and three software versions is considered. Particles with IDs 0-299 belong to hardware platform 1, particles 300-599 belong to hardware platform 2, and particles 600-899 belong to hardware platform 3. Particles with IDs 0-99, 300-399, and 600-699 use software version 1; particles 100-199, 400-499, and 700-799 use software version 2; particles 200-299, 500-599, and 800-899 use software version 3. The resulting nine particle groups are presented in Table 1.

Table 1. Particle Groups with Different Hardware Platforms

	Software version 1	***Software version 2***	***Software version 3***
Hardware platform 1	*h1s1*	*h1s2*	*h1s3*
Hardware platform 2	*h2s1*	*h2s2*	*h2s3*
Hardware platform 3	*h3s1*	*h3s2*	*h3s3*

Thus, during decision-making, we may use not only labels indicating data suspiciousness, but also additional information regarding the degree of particle diversity.

Let us consider the following example. There are six records in the same data set: three contain identical values, and the other three contain identical values but are different from the first group. A problematic situation arises regarding which values

should be considered correct. This is precisely why metadata is required: in one of the groups, the values may carry labels indicating that they are incorrect. These may also be copies of data with the same source ID, although let us assume that such data have already been filtered out. Let us consider the most difficult case, in which all six records originate from different particles, and none is marked with a suspicious or incorrect data bit. In this case, the degree of diversity may be compared: suppose that in the first group the three particle IDs correspond to particles with identical software and hardware, whereas in the second group the particles have different hardware or software platforms. The probability of a related fault in particles with different hardware and software is significantly lower than in identical particles; therefore, the data from the second group are selected.

3. SOFTWARE SIMULATION RESULTS AND DISCUSSION

To verify the proposed theories and algorithms, we developed a simulation environment that extends the software described in [11]. The program simulates the behavior of 900 particles over 1000 iterations. The particles are divided into 9 groups of 100 particles each, corresponding to combinations of three different software versions and three different hardware platforms, denoted as h1s1, h1s2, h1s3, h2s1, h2s2, h2s3, h3s1, h3s2, and h3s3, where h1s2 corresponds to hardware platform 1 and software version 2, etc. At each iteration, a particle "measures" the parameters: pressure, temperature, concentration, and altitude are taken from the input file, and noise is added to all parameters to simulate dispersion of values in the measured field and measurement instrument errors.

At each iteration, the particle performs a failure check, where the probability is calculated according to (3):

$$P_{\text{failure}} = (1 - R_{\text{hw}}) \cdot (1 - R_{\text{sw}}) , \quad (3)$$

where R_{hw} is the reliability of the hardware platform and R_{sw} the reliability of the software version. The reliability values are $R_{\text{hw1}} = 0{,}961$, $R_{\text{hw2}} = 0{,}955$, $R_{\text{hw3}} = 0{,}943$, $R_{\text{sw1}} = 0{,}980$, $R_{\text{sw2}} = 0{,}983$, $R_{\text{sw3}} = 0{,}985$. Thus, the failure probability for group h2s1 is (1 - 0,955) * (1 - 0,980) = 0,0009.

If a particle fails during a given iteration, it no longer participates in the simulation and is considered faulty in all subsequent iterations. Figure 2 presents graphs of the number of operational particles during each of the 1000 simulation iterations. The upper graph shows the total number of operational particles, while the lower section contains nine graphs corresponding to the individual particle groups.

As shown in Figure 2, the expected result is observed: by the end of the simulation, the largest number of operational particles remains in group h1s3, since these particles possess the most reliable hardware platform and software version. However, during the simulation, the curves intersect, demonstrating the significant influence of stochastic events on relatively small groups of particles that otherwise exhibit similar overall reliability.

Each particle has a particle ID ranging from 0 to 899. At each iteration, the availability of communication between particles is checked. The probability depends on

the difference between particle IDs: the closer the IDs, the higher the probability of communication.

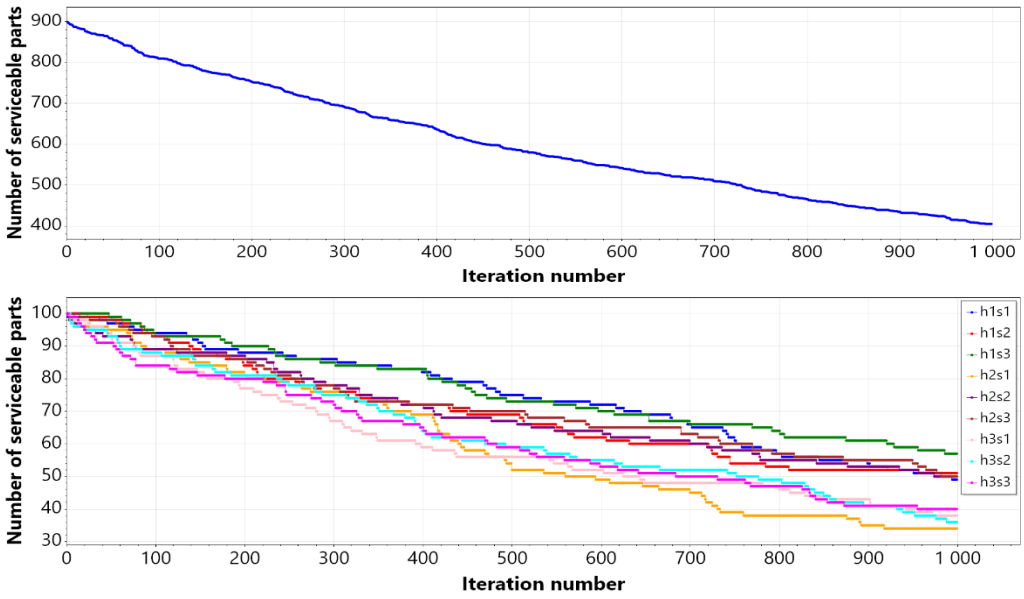


Figure 2. Number of operational particles by iteration

The probability also depends on the distance between particles, which increases quadratically starting from iteration 200. There is a limit on the number of communication links between two particles: a maximum of six communication sessions per iteration. Figure 3 shows the average number of established communication sessions per particle per iteration. The graph indicates that the dominant influence on the number of connections is the distance between particles, whereas the slight decrease during the first 200 iterations is due to particle failures.

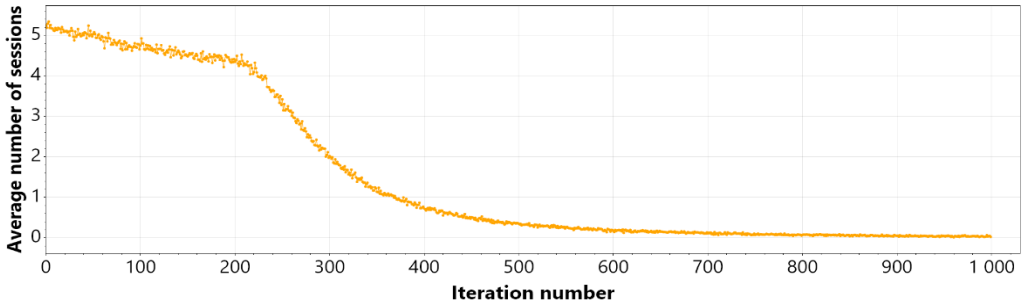


Figure 3. Average number of established communication sessions per particle

Communication verification is performed in two stages. First, it is checked which particles can detect each other (“handshake”). If a particle detects four or more particles, one data set is transmitted; if it detects three particles, two data sets are transmitted; if it detects two neighboring particles, three data sets are transmitted; and if only one

neighboring particle is detected, six data sets are transmitted. However, it must also be verified that the second particle detects the same number of neighbors. For example, if particle A detects only particle B and particle B detects only particle A, then they exchange six data sets: the current-iteration data set and five data sets from previous iterations. However, if particle A detects only particle B, while particle B detects four neighboring particles, then only one data set is exchanged. Thus, the global limitation of six data sets per iteration must be satisfied for each particle.

Figure 4 shows the average number of transmitted data sets. During the first 300 iterations, all particles detect at least 4 neighbors; therefore, exactly one data set is transmitted per iteration. Subsequently, particle dispersion occurs, as a result of which particles detect three or fewer neighbors and switch to transmission modes involving two or more data sets. After iteration 800, situations arise in which all exchanges occur only between two particles, resulting in the transmission of six data sets.

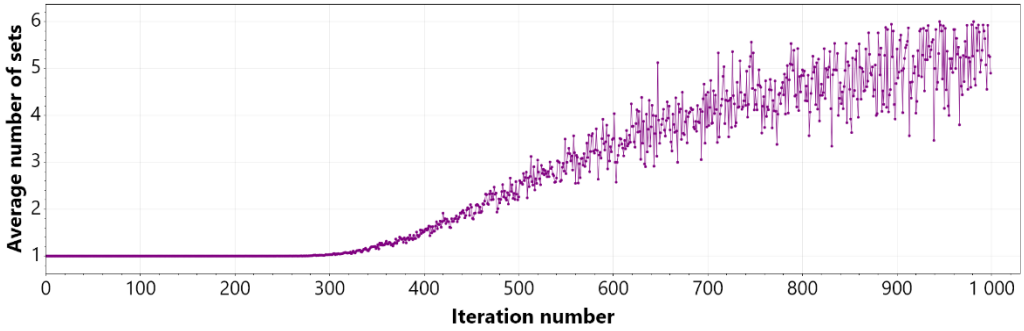


Figure 4. Average number of transmitted data sets

After verifying the communication session, an additional check is performed: each communication session may fail with a certain probability. The probability of communication failure depends on the hardware platform and software version of the communicating parties. If both match, for example particles h2s3 and h2s3 communicate, then the failure probability is 0.02. If only the software differs, the probability is 0.03. If only the hardware platform differs, the probability is 0.04. If both the hardware and software differ, the probability of communication failure is 0.06. If a failure occurs, data exchange does not take place even if the handshake is successful.

Figure 5 presents the number of failed communication sessions. The graph follows the general behavior of the graphs in Figs. 2 and 3, since the failure probability does not depend on the iteration number and decreases as the total number of communication sessions decreases. If a communication session is established between particles, they exchange measured data, and the particles subsequently store not only their own data for the current iteration but also the data of particles with which a communication session has been established. The exchange is bidirectional. The common data storage in the program is implemented as a “table” with the following fields: iteration number, source particle ID (i.e., the ID of the particle that performed the “measurement”), storage ID (i.e., the ID of the particle storing these data; if the particle participated in communication sessions during the iteration, it stores not only its own data but also data

from other particles), and the actual data, namely temperature, pressure, concentration, and altitude. At the end of program execution, this table is written to the output file output.dat.

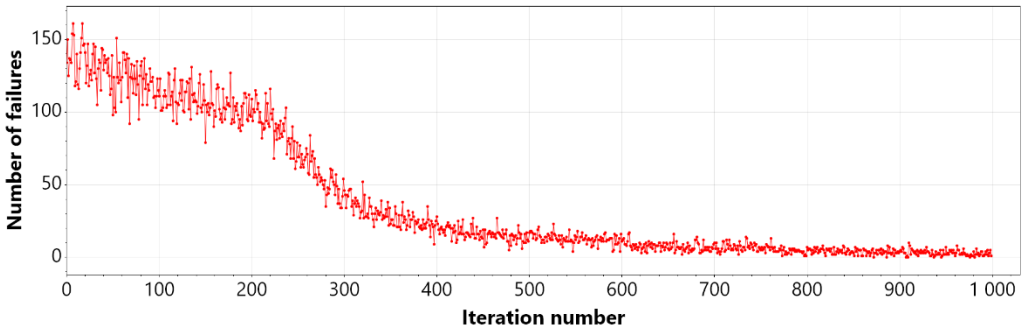


Figure 5. Number of failed communication sessions

The program also maintains a list of particle statuses indicating whether each particle is currently operational. After the simulation, only the data from the common table is retained for which the storage ID corresponds to a particle that remains operational during the final iteration.

Figure 6 presents graphs of the accumulated data for the entire swarm, while Figure 7 shows the corresponding graphs for the particle groups. Of particular interest are the metrics representing the total accumulated information and the amount of unique information. As shown in the diagrams, the total amount of information correlates with the total number of operational particles by group in Figure 3. However, the amount of unique data differs and is determined by the data transmitted from particles that subsequently failed.

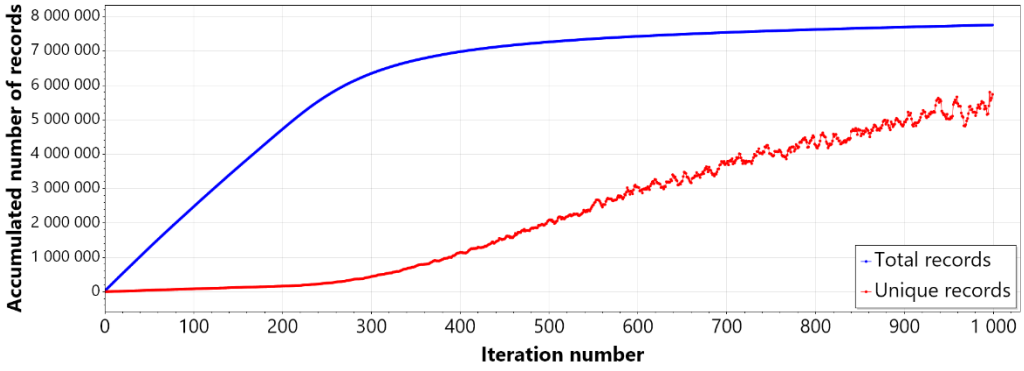


Figure 6. Amount of data accumulated by the swarm

The decreases in the upper red graph are caused by failures of particles whose data copies are not stored by operational particles. When such particles fail, the swarm inevitably loses unique data that can no longer be recovered. However, the overall behavior of the graph shows that the amount of unique data increases, indicating that the swarm preserves more new data than it loses to failed particles. The amount of unique

data originating from particle group h2s3 is nearly equal to that of group h1s3, although fewer operational particles from this group remain (Figure 2). This fact confirms the effectiveness of the proposed data exchange approach between smart dust particles.

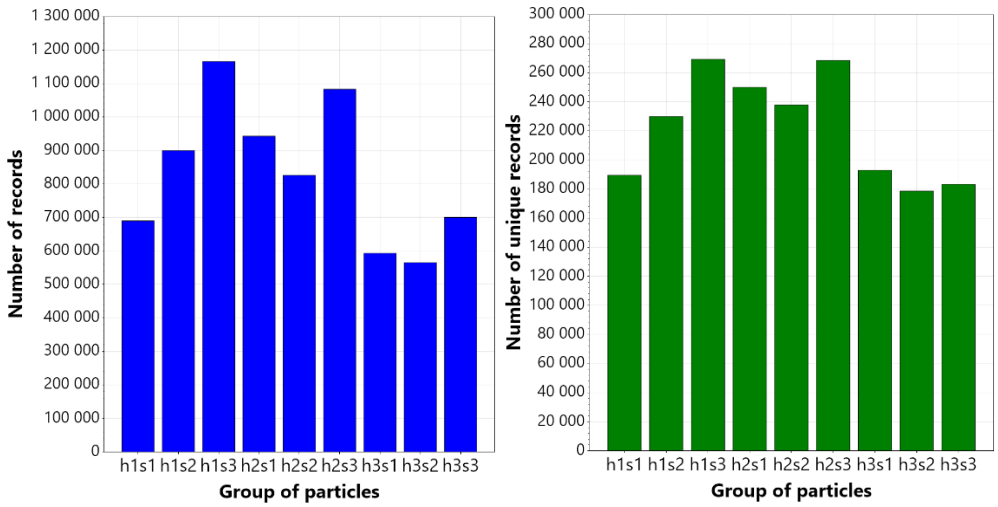


Figure 7. Amount of data accumulated by the particle groups

4. CONCLUSION

In this paper, various methods and algorithms to increase the quantity and reliability of data collected by a smart dust particle swarm under continuous degradation have been proposed. Scenarios of particle behavior during data exchange, depending on their information visibility and bandwidth, have been proposed. Algorithms for labeling potentially unreliable data and for disabling a particle's active role upon repeated transmission of such data have been proposed. An algorithm for resolving collisions in the post-processing of collected data, based on the principle of particle diversity, has been proposed. The proposed algorithms and methods were verified in a simulation environment. The obtained results confirm their operability and effectiveness.

ACKNOWLEDGEMENT

The study was supported by the grant of the Russian Science Foundation No. 25-11-20040, https://rscf.ru/project/25-11-20040/, and by the grant of the Krasnoyarsk Regional Science and Technology Support Fund.

REFERENCES

[1] Tripathi, A.K., Aruna, M., Parida, S., Nandan, D., Elumalai, P.V., Prakash, E., Isaac Joshua Ramesh Lalvani, J.S.C., Rao, K.S. Integrated smart dust monitoring and prediction system for surface mine sites using IoT and machine learning techniques. *Scientific Reports*, vol.14, no.1, 2024, article 7587. DOI: 10.1038/s41598-024-58021-x.

[2] Mondal, I., Haick, H. Smart dust for chemical mapping. *Advanced Materials*, vol.37, no.19, 2025, article e2419052. DOI: 10.1002/adma.202419052.

[3] Gao, G., Feng, Z., Han, T. Data Authentication for wireless sensor networks with high detection efficiency based on reversible watermarking. *Wireless Communications and Mobile Computing*, vol.2021, 2021, article 6651137, 13 p. DOI: 10.1155/2021/6651137.

[4] Kanwal, T., Altaf, S., Javed, M.K. Environmental monitoring smart system with self-sustaining wireless sensor network using data validation algorithms. *Journal of Engineering Sciences*, vol.7, no.1, 2020, pp. E10-E19. DOI: 10.21272/jes.2020.7(1).e3.

[5] Liu, Z., Zhang, J., Liu, Y., Feng, F., Liu, Y. Data aggregation algorithm for wireless sensor networks with different initial energy of nodes. *PeerJ Computer Science*, vol.10, 2024, article e1932. DOI: 10.7717/peerj-cs.1932.

[6] Bandealinaeini, H., Salehkaleybar, S. Broadcast distributed voting algorithm in population protocols. *IET Signal Processing*, vol.14, 2020, pp. 846-853. DOI: 10.1049/iet-spr.2020.0297.

[7] Katenka, N., Levina, E., Michailidis, G. Local vote decision fusion for target detection in wireless sensor networks. *IEEE Transactions on Signal Processing*, vol.56, no.1, 2008, pp. 329-338. DOI: 10.1109/TSP.2007.900165

[8] Tomic, S., Beko, M., Vukobratovic, D., Krco, S. Voting scheme to strengthen localization security in randomly deployed wireless sensor networks. *arXiv preprint*, arXiv:2502.20218 [eess.SP], 2025. DOI: 10.48550/arXiv.2502.20218.

[9] Rajesh, D., Rajanna, G.S. Energy aware data harvesting strategy based on optimal node selection for extended network lifecycle in smart dust. *Journal of Intelligent & Fuzzy Systems*: *Applications in Engineering and Technology*, vol.44, no.1, 2023, pp. 939-949. DOI: 10.3233/JIFS-221719.

[10] Chéour, R., Jmal, M.W., Khriji, S., El Houssaini, D., Trigona, C., Abid, M., Kanoun, O. Towards hybrid energy-efficient power management in wireless sensor networks. *Sensors*, vol.22, no.1, 2022, article 301. DOI: 10.3390/s22010301.

[11] Kovalev, I.V., Losev, V.V., Saramud, M.V. Formalized method of inter-machine interaction of intelligent particles with a series of behavioural scenarios of particles and the swarm as a whole. *International Journal on Information Technologies and Security*, vol.17, no.4, 2025, pp. 45-56. DOI: 10.59035/TVDK7571.

Information about the authors:

Igor Vladimirovich Kovalev – Professor at the Department of Software Engineering, Siberian Federal University and Professor at the Department of systems analysis and operations research, Reshetnev Siberian State University of Science and Technology. Areas of Research are fault-tolerant software, system analysis, software engineering.

Vasiliy Vladimirovich Losev – Associate Professor at the Department of automation of production processes, Reshetnev Siberian State University of Science and Technology. Areas of Research are automation control systems, system analysis.

Mikhail Vladimirovich Saramud - Senior Researcher of research department, Reshetnev Siberian State University of Science and Technology. Areas of Research are fault-tolerant software, system analysis.

Manuscript received on 01 June 2026