RASS: Risk-Aware Swarm Storage

Extended Abstract

Samuel Arseneault
Polytechnique Montréal
Montréal, Canada
samuel.arseneault@polymtl.ca

David Vielfaure
Polytechnique Montréal
Montréal, Canada
david.vielfaure@polymtl.ca

Giovanni Beltrame
Polytechnique Montréal
Montréal, Canada
giovanni.beltrame@polymtl.ca

ABSTRACT

In robotics, data acquisition often plays a key part in unknown environment exploration. For example, storing information about the topography of the explored terrain can inform the decision-making process of the robots. Therefore, it is crucial to store these data safely and to make it available quickly to the operators of the robotic system. In a decentralized system like a swarm of robots, this entails several challenges. To address them, we propose RASS, a decentralized risk-aware swarm storage and routing mechanism, which relies exclusively on local information sharing between neighbours to establish storage and routing fitness. We test our system through thorough experiments in a physics-based simulator and obtain convincing reliability, routing speeds, and swarm storage capacity results.

KEYWORDS

Swarm Robotics; Information Sharing; Robot Safety

ACM Reference Format:
Samuel Arseneault, David Vielfaure, and Giovanni Beltrame. 2022. RASS: Risk-Aware Swarm Storage: Extended Abstract. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 3 pages.

1 INTRODUCTION

Using multi-robot systems for the exploration of unknown environments is very appealing. Indeed, if robots do not overlap in their exploration task, the amount of terrain covered increases proportionally with the number of robots in the system [4]. This can, for example, be particularly useful for search and rescue scenarios [6] where the speed at which the environment is covered is of critical importance. However, as the number of robots increases so does the amount of data collected, which puts pressure on the data storage infrastructure. Unfortunately, multi-robot systems usually suffer from unreliable connectivity [2], and directly sending the information to an external storage system (e.g., the cloud) may not always be feasible. Furthermore, the robots in a swarm are not necessarily reliable [13]: even in controlled environments, they are usually meant to be easily replaced. This issue is aggravated when they must face situations that decrease their reliability, such as exposure to dangerous environments. Therefore, giving the robots a way to eventually relay the information they acquired during their mission to a control station for more permanent storage is a definite advantage: it not only allows the information to be stored in a safe location accessible by human operators, it also alleviates the memory usage of the robots and enables continuous operation. Nonetheless, because in practice robots have a finite communication range, the information collected at the periphery of the swarm usually needs to be routed through other robots before reaching a base station. These constraints motivate the need for a completely decentralized, risk-aware swarm storage system that can safely and efficiently store data and route it towards a base station in a percolating fashion whenever possible.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 SYSTEM MODEL

We consider a fully decentralized multi-robot system tasked with the exploration of a dangerous environment. The multi-robot system is denoted as the collection of agents $a_i \in A$. We assume that the swarm can be represented by a *connected* graph $\mathcal{G}$ with nodes $\mathcal{A}$ and edges $\mathcal{L}$ respectively representing agents and their wireless communication links. In practice, this connectivity cannot be maintained at all times, because of the quality of the links in $\mathcal{L}$ and because of the movements of nodes $\mathcal{A}$. We consider the amount of data needed to be stored to be greater than the individual storage capabilities of the robots. Also, we consider that the robots are only able to communicate if they are within a certain communication radius R.

Radiation is known to cause performance loss and failures in robots [8, 11]. We therefore adapt the risk modelling from [12], which is based on a set of independent point radiation sources S with individual intensity $I_j \sim \mathcal{U}(0,1)$. The perceived intensity decays exponentially (with λ as a decay parameter) as the Euclidean distance $\rho(\boldsymbol{x}_i)$ between $\boldsymbol{s}_j$ and $\boldsymbol{x}_i$ increases. The total perceived radiation level by a robot a_i at position $\boldsymbol{x}_i \in E$ is given by:

$$r(\boldsymbol{x}_i) = b + \sum_{s_j \in S} \frac{I_{s_j}}{1 + \lambda \rho(\boldsymbol{x}_i)^2} \tag{1}$$

and is measured by an on-board sensor with Gaussian measurement noise $b \sim \mathcal{N}(0, 0.05)$. We posit the radiation's effect on the system is to cause data corruption [8]. The probability of a datum being corrupted due to the combined effect of radiation sources follows a Bernoulli distribution given by:

$$\mathbb{P}(c_i = 1|S) \sim \mathcal{B}(r(\boldsymbol{x}_i)) = 1 - \prod_{s_j \in S} 1 - \mathbb{P}(c_i = 1|\boldsymbol{s}_j) \tag{2}$$

Our risk-aware storage system draws inspiration from SwarmMesh [7], in that each node periodically assigns itself a potential ϕ_i based on its fitness to store data, given by:

$$\phi_i = \begin{cases} \frac{1}{\alpha h_i + \beta r(\boldsymbol{x}_i)} & \text{if } m_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where m_i is the memory available on node i, r_i is the risk associated with the current node's location (stored in the distributed belief map) and h_i refers to the minimum hop count required to reach the base station from i as specified in the routing table. Given the existence of at least one (multi-hop) path between any node and the base station, we can establish a routing table based on hop count as suggested by [1]. Parameters α and β are respectively the routing weights and risk weights, which allow adapting the policy based on the relative importance of the routing time and the environmental risk with respect to each other. Similarly to [7], a node which becomes unfit to store data will evict such data (using a Least Recently Used policy) by moving it into its routing queue. The condition for "unfitness" is simply:

$$T\phi_i < \max_{j\in\mathcal{N}} \phi_j \tag{4}$$

where $\mathcal{N}$ and T are the set of i's neighbours and the fitness threshold, respectively.

3 EXPERIMENTS

We ran extensive simulations in a physics-based simulator, ARGoS [10] with models of Khepera IV [5] robots. We executed 30 simulation runs with 100 robots for each type of experiment to reach results with low uncertainty. The robots are placed within a 20m by 20m arena and their communication radius R is set to 3m. Three radiation sources are randomly distributed in the environment around the origin. The base station is located in a corner of the arena and its storage capacity is assumed to be infinite. In order to replicate realistic operation scenarios, we artificially introduce bandwidth limitations of 10 data items transfered per robot at every time step. For the same reason, our simulated robots have a limited storage capacity of 50 data items of at most 50 bytes each. This gives a total storage capacity of 2500kB per robot.

To evaluate the performance of our system in different scenarios, we tested it with static topologies: a grid-like formation, and a scale-free network; as well as with dynamic topologies: a formation obtained through Lennard-Jones potential interactions and a formation evolving from random walk motions. Our first benchmark algorithm is to use a fitness policy based purely on hop count, i.e. if required, data is sent only to neighbours closer to the base station. Our second comparison baseline is to store data in a virtual stigmergy [9]. The first metric we used in our performance evaluation is the average data transfer speed, measured as the delay between the creation of the datum and its arrival by percolation to the base station. We excluded results of this metric for the virtual stigmergy, as the stigmergy cannot include the concept of a base station (since all nodes are peers), and stigmergy propagation speeds are detailed in [9]. The second metric we used is reliability, expressed as $\frac{n_g - n_l}{n_g}$, where n_g and n_l are respectively the amount of data generated and the amount of data lost at the end of the simulation.

The results obtained in the 30 simulation runs for the static topologies (grid-like and scale-free) as well as for the dynamic topologies (Lennard-Jones potential and random walk) are presented in table 1. Results show that RASS outperforms the hop-count algorithm in terms of reliability. Because of the risk awareness component included in its fitness policy as detailed in Eq. 3, robots do not always route the data through the shortest path to the base station. RASS avoids the dangerous storage nodes of the system when routing data which explains the higher reliability levels achieved. This is why, on average, RASS takes 54.89% more time to route the data to the base station when compared to the hop-count algorithm. On the other hand, the hop-count algorithm always takes the shortest path towards the base station regardless of the risk associated with it. This leads to a higher number of data losses due to corruption and an overall lower reliability. However, hop count can yield faster transfer speeds as shown in table 1. For the virtual stigmergy, most of the data losses can be attributed to the storage having reached its maximum capacity. Indeed, because of the fully replicated nature of the stigmergy, the memory of the agents is quickly saturated. This full redundancy prevents losing data from corruptions, however it entails a very inefficient use of the memory of the robots and ultimately leads to data losses due to insufficient memory capacity.

Table 1: Average transfer speed and average individual memory usage with different topologies

Topology	*Algorithm*	*Transfer speed (hops)*	*Reliability (%)*
Grid-like	RASS	11.45	77.5
	Hop-Count	9.11	46.6
	Stigmergy	N.A.	1.9
Scale Free	RASS	11.44	82.1
	Hop-Count	6.85	72.9
	Stigmergy	N.A.	1.9
Lennard-Jones	RASS	12.51	90.2
	Hop-Count	7.32	66.5
	Stigmergy	N.A.	1.8
Random search	RASS	12.68	87.4
	Hop-Count	7.76	64.8
	Stigmergy	N.A.	2.0

4 CONCLUSIONS

We presented RASS, a Risk-Aware Swarm Storage system in which a swarm of robots can collectively store data on strategically chosen members. This choice is made without central coordination and is purely based on local information shared between the robots. This information is simply composed of risk measurements and topological distance from a robot to a base station, and a used to determine a robot's fitness to store data as well as to establish the most reliable and fast route towards the base station. We show in our experiments that RASS largely outperforms a hop-count-based solution as well as a virtual stigmergy in terms of reliability while only being slightly slower in terms of percolation speed compared to the hop-count-based algorithm. RASS showed good scalability in physics-based experiments as it repeatedly performed well with a large number of robots. The complete details regarding RASS' implementation, some relevant related works and physical experiments showcasing the real-world applicability of our system are presented in [3].

REFERENCES

[1] Md Ibrahim Abdullah, Mohammad Muntasir Rahman, Mukul Chandra Roy, et al. 2015. Detecting sinkhole attacks in wireless sensor network using hop count. *IJ Computer Network and Information Security* 3 (2015), 50–56. http://www.mecs-press.org/ijcnis/ijcnis-v7-n3/v7n3-7.html

[2] Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. 2017. Multirobot exploration of communication-restricted environments: A survey. *IEEE Intelligent Systems* 32, 6 (2017), 48–57. https://ieeexplore.ieee.org/abstract/document/8267592

[3] Samuel Arseneault, David Vielfaure, and Giovanni Beltrame. 2022. RASS: Risk-Aware Swarm Storage. arXiv:2201.01349 [cs.RO]

[4] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. 2005. Coordinated multi-robot exploration. *IEEE Transactions on robotics* 21, 3 (2005), 376–386. https://ieeexplore.ieee.org/abstract/document/1435481

[5] K-Team. 2021. Khepera IV. https://www.k-team.com/khepera-iv

[6] George Kantor, Sanjiv Singh, Ronald Peterson, Daniela Rus, Aveek Das, Vijay Kumar, Guilherme Pereira, and John Spletzer. 2003. Distributed Search and Rescue with Robot and Sensor Teams. *Springer Tracts in Advanced Robotics* 24, 529–538. https://doi.org/10.1007/10991459_51

[7] Nathalie Majcherczyk and Carlo Pinciroli. 2020. SwarmMesh: A Distributed Data Structure for Cooperative Multi-Robot Applications. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4059–4065. https://ieeexplore.ieee.org/abstract/document/9197403

[8] George C Messenger and Milton S Ash. 1986. The effects of radiation on electronic systems. (1986). https://inis.iaea.org/search/search.aspx?orig_q=RN:18091073

[9] Carlo Pinciroli, Adam Lee-Brown, and Giovanni Beltrame. 2016. A tuple space for data sharing in robot swarms. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. 287–294. https://carlo.pinciroli.net/pdf/Pinciroli:BICT2015.pdf

[10] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. 2012. ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intelligence* 6, 4 (2012), 271–295.

[11] Richard Sharp and Marc Decreton. 1996. Radiation tolerance of components and materials in nuclear robot applications. *Reliability Engineering & System Safety* 53, 3 (1996), 291–299. https://doi.org/10.1016/S0951-8320(96)00054-3

[12] David Vielfaure, Samuel Arseneault, Pierre-Yves Lajoie, and Giovanni Beltrame. 2021. DORA: Distributed Online Risk-Aware Explorer. arXiv:2109.14551 [cs.RO] https://arxiv.org/abs/2109.14551

[13] Alan FT Winfield and Julien Nembrini. 2006. Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control* 1, 1 (2006), 30–37. https://www.inderscienceonline.com/doi/abs/10.1504/IJMIC.2006.008645