Coordinating a Swarm of Micro-Robots Under Lossy Communication

Razanne Abu-Aisheh, Francesco Bronzino, Myriana Rifai, Lou Salaun, Thomas Watteyne

Coordinating a Swarm of Micro-Robots Under Lossy Communication

Razanne Abu-Aisheh
razanne.abu-aisheh@nokia.com
Nokia Bell Labs and Inria
Paris, France

Francesco Bronzino
fbronzino@univ-smb.fr
LISTIC, Université Savoie Mont Blanc
Chambery, France

Myriana Rifai
Lou Salaun
myriana.rifai@nokia-bell-labs.com
lou.salaun@nokia-bell-labs.com
Nokia Bell Labs
Paris, France

Thomas Watteyne
thomas.watteyne@inria.fr
Inria
Paris, France

ABSTRACT

We envision swarms of mm-scale micro-robots to be able to carry out critical missions such as exploration and mapping for hazard detection and search and rescue. These missions share the need to reach full coverage of the explorable space and build a complete map of the environment. To minimize completion time, robots in the swarm must be able to exchange information about the environment with each other. However, communication between swarm members is often assumed to be perfect, an assumption that does not reflect real-world conditions, where impairments can affect the Packet Delivery Ratio (PDR) of the wireless links. This paper studies how communication impairments can have a drastic impact on the performance of a robotic swarm. We present Atlas 2.0, an exploration algorithm that natively takes packet loss into account. We simulate the effect of various PDRs on robotic swarm exploration and mapping in three different scenarios. Our results show that the time it takes to complete the mapping mission increases significantly as the PDR decreases: on average, halving the PDR triples the time it takes to complete mapping. We emphasise the importance of considering methods to compensate for the delay caused by lossy communication when designing and implementing algorithms for robotics swarm coordination.

1 INTRODUCTION

Exploration of post-disaster environments for target detection is a risky, time and resource consuming mission. We envision autonomous swarms of micro-robots to be tremendously beneficial in terms of minimizing exploration time and reducing human exposure to risks [12]. A swarm of micro-robots is defined as a group of at least three robotic entities that cooperate together to achieve a common global goal with limited to zero human operated control. During these operations, micro-robots strategically search the environment to collect the most informative data from their surroundings. Thus, the ability to leverage the collected data to the benefit of other members of the swarm is critical to minimizing search completion time.

The goal of a exploration expedition is for the swarm to map the entire environment, leaving no accessible parts unexplored [8]. The completeness of robot-built maps, as well as the speed at which this is accomplished, are major challenges [13]. Previous work has strongly focused on minimizing completion time over other metrics due to the urgency of the missions.

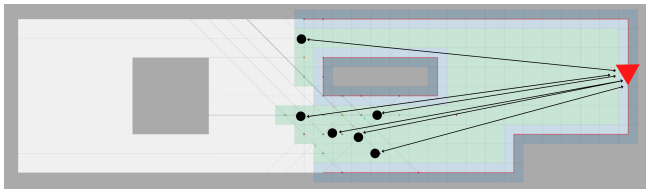

Figure 1: The orchestrator and robotic swarm in an environment, showing a partially built map of a previously unknown environment.

Typically, swarm robots wirelessly communicate information about the environment they are exploring to a central "orchestrator". Having centralized situational awareness at an orchestrator is often required for the effective supervision of the mission [5]. The study of this problem is often simplified by assuming that communication between robots is possible between any two locations. However, such a strong assumption does not necessarily reflect real-life conditions where imperfections in the communication channel can impact the performance of the system [3].

Our previously published Atlas [1] is an algorithm which runs at the orchestrator and coordinates the action of individual robots in the swarm as they carry out a mapping expedition. Atlas minimizes the time to fully map an unexplored area, and the number of micro-robots necessary to complete the mapping. While Atlas does not require continuous communication between all the members of the swarm, it assumes ideal communication: there are no packet losses or other limitations regarding communication.

We use Atlas as a representative swarm coordination algorithm, to evaluate the impact of network connectivity on mapping algorithms. Specifically, we study the impact of lossy communication on the speed of the mapping by Atlas. We then modify Atlas so it handles communication failures while maintaining the guarantee of mapping completion. We develop and use a discrete-event continuous time simulator that includes realistic communication

conditions to evaluate the completion time even for extremely lossy environments.

This paper aims at demonstrating the importance of considering communication disturbances and losses when designing swarm cooperation algorithms for critical exploration based missions. The contributions of this paper are threefold:

- We develop a discrete-event, continuous-time and -space simulator that integrates lossy communication models.
- We design a modified version of the Atlas algorithm to include packet loss tolerant exploration and mapping that guarantees a 100% completion ratio.
- We emphasize the need for focusing on communication limitations when designing exploration algorithms by signifying the effect of packet loss on mission time-to-completion.

The remainder of this paper is organized as follows. Section 2 surveys related work. Section 3 presents our system model and the challenge we address in this paper. Section 4 describes the communication model and its implementation in the simulator. Section 5 details the modifications made to the Atlas algorithm. Section 6.1 describes the implementation of Atlas in a simulator, taking the work by Abu-Aisheh *et al.* [1] as a starting point. Section 6.2 showcases the impact of packet loss on the performance of Atlas. Finally, Section 7 concludes this paper.

2 RELATED WORK

Robotic swarm exploration is often simplified by assuming that communication between robots is always possible between any two locations without packet loss [8]. Here, we survey the related work that *does* consider lossy connectivity.

Amigoni *et al.* [2] categorize communication for robotic swarm exploration to three categories: robots are *not* required to communicate, every robot must be able to communicate with all other robots at all times, or communication is only required either periodically or by particular events such as the discovery of new information.

Manfredi *et al.* [10] propose an algorithm tolerant to packet loss. The network of robots is composed of one leader and several followers. The goal of the algorithm is to set the position, velocity, and control parameters of the followers in a manner that enables them to follow the leader. Depending on the rate of packet loss, control inputs are corrected to reduce the error. While this algorithm is packet-loss tolerant, it purely depends on maintaining a close distance with other robots, as it assumes the existence of a joint path from the leader to every follower across each uniformly bounded interval.

Benavides *et al.* [6] also propose an exploration strategy for multi-robot systems. Their approach consists in avoiding disconnection between the robots by having the robots be aware of the connectivity. Given the position of robots and obstacles, robots estimate the connectivity degree of a specific location. Robots can only confirm the absence of connectivity or deliver an optimistic estimation of connectivity. This is equivalent to either having a 100% Packet Delivery Ratio (PDR) if the robots can connect, or 0% if not. Data losses in between are not taken into account here.

Banfi *et al.* [4] propose the concept of "recurrent connectivity", where planning is centralized and robots connect back to the central orchestrator only when a new piece of information is available. This eliminates the need for a full communication graph. Packet loss is not directly considered here: robots are assumed to be able to communicate in line-of-sight conditions.

Few research has considered the effect of packet delivery ratio on the overall performance of exploration and mapping. Zhivkov *et al.* [14] examine the impact of degrading communication quality in a swarm with the aim of quantifying the effects and assessing the risks associated with poor communication quality in robotic swarms. To do so, they conduct a series of experiments with multiple message transmission success rates. They do this using simulation and experimentation. Their simulation results show that the exploration time increases as the packet loss percentage increases. However, the increase in exploration completion time from when no packet loss is not experienced to that of when it is 75%, is only 10 s in simulation, while it is around 40 s in the physical experimentation. While this added time to mapping completion seems insignificant, the paper does not provide any details about the exploration or communication algorithms used, and focuses on random packet loss rates that do not align with real life packet losses. It is therefore difficult to infer from their evaluation the performance of the exploration algorithm, or the severity of the impact of packet losses, in time critical exploration scenarios.

Jensen *et al.* [8] discuss how communication impacts online multi-robot coverage algorithms' viability for real-world scenarios, and show how communication can affect the performance of various algorithms. However, they focus more on comparing different algorithms with different communication models, as opposed to comparing degrading communications and losses with the same exploration algorithm and communication model.

Similar to [14], we evaluate the effect of packet loss on the performance of exploration and mapping. We then develop a communication and exploration algorithm that is robust to degrading network conditions. We evaluate the performance of our mapping solution for different PDR values. We design an event-based communication protocol where the robots in a swarm communicate with a central orchestrator once triggered by specific events.

3 SYSTEM MODEL AND CHALLENGE

Our system model consists of the following elements:

(1) **Robots.** We assume each micro-robot is small enough that it can be modeled as a dot with (x,y) coordinates. We also assume that each robot can move at a speed of up to 1 m/s, has sensing capabilities limited to a bump sensor that is triggered upon contact with an obstacle, and the ability to wirelessly communicate. We went for basic robots with minimal capabilities in order to reduce size and cost significantly; making obtaining and maintaining large swarms of robots more feasible.
(2) **Orchestrator.** This central entity is responsible for coordinating the exploration by the robots. The centralized nature of the orchestrator enables better exploration strategies based on its global view.
(3) **Environment and communication.** The environment is initially unknown to the system. All robots start the exploration from the location of the orchestrator. The robots only report

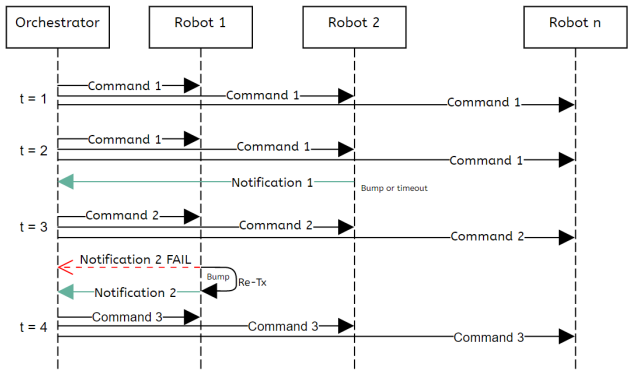

Figure 2: Communication between the orchestrator and the swarm.

back to the controller when either of two possible events happen: a) a robot's bump sensor is triggered, or b) a robots assigned moving duration timer runs out. In this manner, the swarm behaviour is asynchronous as the orchestrator updates the movement plan for that particular robot only when it hears back from it. We assume communication limitation by modelling packet loss into the environment in order to represent more realistic losses in a typical environment.

(4) **Exploration and mapping.** We use the Atlas algorithm [1] as a starting point, which we modify to be tolerant to packet loss. We refer to this version as "Atlas 2.0". The mapping is represented by dots with (x,y) coordinates on a continuous map, where each dot represents the location at which a robot's bump sensor was triggered. These dots connect into lines once they are a certain distance apart and create an outline of all the obstacles in an environment, see Section 5.

We focus on developing and validating a mapping algorithm that reliably completes all of the time, even with packet loss. We evaluate the impact of packet loss on the time it takes to complete the mapping task.

4 COMMUNICATION PROTOCOL

We design a communication protocol that takes packet losses into account to guarantee mapping completion with any PDR above zero. In the protocol, communications occur between the orchestrator and each robot in a swarm (and vice-versa) through two types of packets: commands (from the orchestrator to robots) and notifications (from robots to the orchestrator). The protocol is based on an event-based communication model with recurrent connectivity requirements. That is, robots only communicate back to the orchestrator when they have new relevant data or have reached the assigned target position they were directed to go to. Otherwise, no connectivity is needed between the robots and the orchestrator. Time is cut into 1 s cycles, with two steps in each cycle.

Step 1. Commands. The orchestrator transmits a packet. This command packet contains the heading, speed, and movement duration for each robot in the swarm (Fig. 3). Headings refer to the direction a robot should take, between 0 and 360 degrees. Movement duration is the amount of time the robot should move for. Note

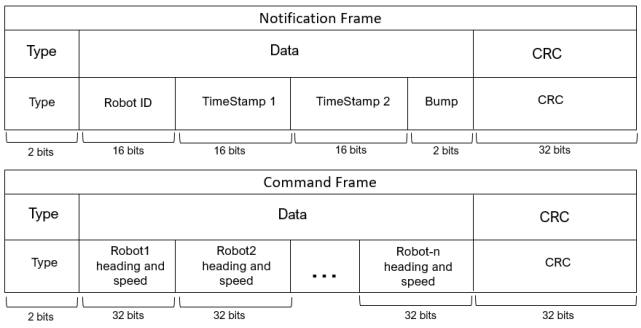

Figure 3: Packet frames for commands and notifications

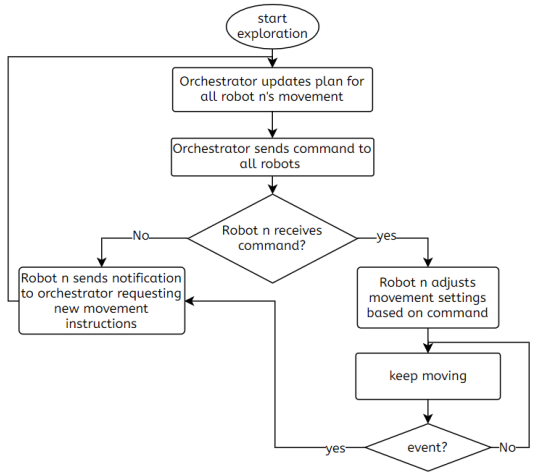

Figure 4: A flow chart representing the communication between each robot and the orchestrator

that broadcasting was chosen as a means for communication to avoid the need for complex routing tables when addressing specific robots in large swarms.

Step 2. Notifications. When a command is broadcast, all robots that do receive the command packet check if their instructions of movement have been updated. If not, they ignore the command and continue moving according to their previous instructions. Otherwise, they extract their next heading, movement duration and speed, and start moving. The robots keep moving in the given direction, without the need for connectivity, until an event happens, at which point they stop moving. When an event occurs, a robot transmits a notification packet to the orchestrator. The notification packet contains four pieces of information (Fig. 3): the robot ID, a first timestamp with the time at which the robot started moving, a second timestamp with the time at which the robot stopped moving and sent the notification, and whether or not the robot had bumped upon stopping. The logic behind using two timestamps will be explained further on in this section.

As shown in Figs. 2 and 4, the protocol tolerates packet loss by incorporating the following logic: In terms of commands, the headings for each robot are only updated when the orchestrator receives

a notification from the robot. These commands are transmitted periodically every second. Robots that are still moving don't need to "listen" as they haven't bumped or reached their target, and are safe to move in the same direction. The only new information in a command will be relevant to the robots that notified the orchestrator in request for a new heading. As for notifications, when a robot sends a notification, it waits to get a command back from the orchestrator pointing it in a certain direction. If either command or notification packets are lost and the robot doesn't hear back from the orchestrator with a new command, it keeps re-transmitting the notification every second, until it receives a new command. Hence the need for two timestamps. Without packet loss (PDR = 1), one timestamp would suffice as the robot would report the current time as the event time and the orchestrator would receive it on time. Given that the orchestrator already knows the speed and headings of all robots, as well as when and where they last stopped, when it hears back from a robot it takes the stop time and back traces the location of the robot. However, when packets are lost and notifications are re-transmitted, there is a gap in time between the last time the orchestrator recorded an event, when the robot actually started moving after stopping, and when the new event occurred. With two timestamps, the orchestrator knows exactly when the robot started moving from one timestamp, and exactly when it stopped from the other, and can therefore accurately calculate the new position at which the robot stopped. Further details on the mapping are explained in Section 5.

5 EXPLORATION AND MAPPING

5.1 Exploration

Atlas is an exploration algorithm by Abu-Aisheh *et al.* [1], designed for sparse robot swarms. Because it is centralized, it relies on robust communication between each robot and the "orchestrator". We choose Atlas as a starting point as any packet loss causes the exploration and mapping expedition to fail. It uses frontier-based systematic exploration: robots are controlled by a central orchestrator which maintains a partial map throughout the exploration and sends robots to explore the yet unexplored zones within the area. However, in that version of Atlas, ideal lossless communication is assumed. We modify the previous version of Atlas to make it tolerant to lossy communications; we call that Atlas 2.0.

Atlas is synchronous: all robots start moving at the same time and stop moving at the same time. Atlas 2.0 is asynchronous: each robot receives a new command with movement instructions every time it has an event occur. This is done to reduce the overall time it takes to complete mapping and to reduce the impact of packet loss. If one robot is stuck and hasn't received a new packet, it does not affect the rest of the swarm.

The orchestrator maintains an artificial overlay grid that it builds on the go during the exploration which can be expanded infinitely. Each grid cell belongs to one of the following categories at every point in time:

- *Open Cells (OC)*: containing no obstacles
- *Obstacle Cells (ObC)*: containing obstacles
- *Unexplored Cells (UC)*: cells that have been built during the exploration and navigation process but have not yet been explored.

Algorithm 1: Setting new movement instructions in Atlas 2.0

```
OCs = empty;
ObCs = empty;
frontier_cells = empty;
Back track all cells traversed by robot;
OCs ← traversed_cells ;
if Robot bumped then
    ObCs ← current_cell;
    Add "dot" to map at robot position;
end
if current_cell ∈ OCs & connected to UC then
    frontier_cells ← current_cell;
else
    Find closest frontiers to robots;
    Find closest frontier to start point;
    frontier_cells ← selected_frontier;
end
Choose random UC connected to selected_frontier;
Set chosen cell as target;
Choose shortest path to target via A* algorithm;
Use vectoring to set next heading,
  speed,movement duration;
Update command;
```

A robot stops and sends a notification to the orchestrator upon the occurrence of either of two evens: *(1)* it bumped into an obstacle and its bump sensor got triggered, *(2)* it reached the target unexplored cell the orchestrator assigned it, indicated by its movement duration timer running out. Once the orchestrator receives a notification from a robot, it sends new movement instructions to that robot.

If we apply Algorithm 1 starting from the first cell and taking that as the *current cell* for robot *n*, as an example, the next movement instructions would be set as follows. Given that the robot is already inside that cell, the starting cell is an open cell. Since there are no explored cells yet, the starting cell is considered a frontier cell. Note that, if we have multiple frontier cell options, the one closest to the starting cell will be chosen.

All overlay cells directly connected to this frontier – i.e. are within its direct surrounding neighbours without having to pass any other cells to reach it – are valid targets. A random cell out of these is selected as the target for this robot. The A* algorithm [7] is used to find the shortest path to that target. Vectoring is used to set the new movement instruction for the robot. Vectoring is a navigation service provided to aircraft by air traffic control: the controller decides on a particular airfield traffic pattern for the aircraft to fly, the aircraft follows this pattern when the controller instructs the pilot to fly specific headings at appropriate times. In Atlas 2.0, the orchestrator replaces the controller and the robot replaces the plane. The movement pattern is the path generated by A*. A robot moves at the speed and in the heading instructed by the orchestrator until its allocated movement duration runs out, unless it bumps into an obstacle.

The overall behavior is that the frontier expands "away" from the starting position: the robots are controlled to "push" the frontier further from the starting point. In scenarios where there are many obstacles, the swarm can be cut into subgroups as it navigates around obstacles.

5.2 Mapping

The map represents an outline of the walls and obstacles in a bounded unknown environment, with the assumption that obstacles can be broken down into square shaped basic elements. These basic elements are referred to as minimum obstacle features. The orchestrator initiates exploration by sending a command containing the headings h, speeds s and movement duration (after which it should stop to change its heading and redirect itself towards the target) for all the robots in the swarm. The orchestrator stores the initial positions of the robots, as well as the headings and speeds sent for each robot per command. Once a robot bumps into an obstacle, it stops moving and reports the time Tm it started moving and the time Tb it bumped back to the orchestrator. It also does this when the movement duration times out and a heading update is due. In this case, however, the packet indicates that no bump occurred, in order to avoid adding data to the map. The orchestrator calculates the position of the robot it just received a notification from, based on (1) and (2). The orchestrator then updates the next command to include a new heading, speed and movement duration for that robot. It also updates the last known position of this robot.

$$newx = (Tb - Tm) \times \cos h \times s \tag{1}$$

$$newy = (Tb - Tm) \times \sin h \times s \tag{2}$$

Every bump is stored as a "dot" on the map, representing the (x,y) coordinates of the robot at which its bump sensor was triggered. Any two dots are connected into a line if the distance between them is less than the size of the minimum obstacle size. This is because the two dots are on an obstacle and an obstacle can not be smaller than that size. Any common points on two lines lead to the two lines being connected at that point in the same method. Mapping completion is detected once a line "loop is closed": it has no disconnected edges. The map builder constantly checks all edges to see if they can be connected to one another.

6 EXPERIMENTAL RESULTS

6.1 Simulation Platform

Atlas comes with an open-source simulator, developed by Abu-Aisheh *et al.* for comparing exploration and mapping algorithms. In this paper, we build upon that simulator and improve it for the purpose of investigating the effect of packet loss on event-based communication models[1]. Fig. 1 is a screenshot of the user interface of the simulator.

In the simulator, we represent a 2D space in a continuous manner. The simulator is discrete-event: time updates as scheduled events are processed, leading to a continuous time representation. We assume each robot has a "bump" sensor that get triggered whenever it hits an obstacle.

[1] As an online addition to this paper, the simulator is published under and open-source license at https://github.com/openwsn-berkeley/Atlas.

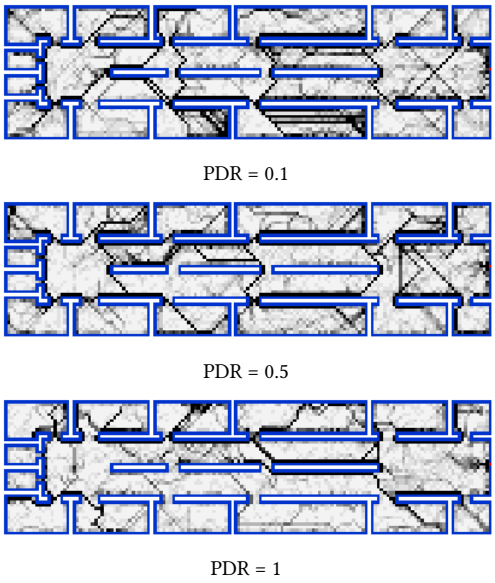

Figure 5: Heat maps of how often robots have been present on each cell, at the end of a simulation run. Results presented for a 50-robot swarm. The darkest cells represent cells that have been passed by 10 or more times. The floorplan used for the simulations had a size of (80×21 cells). The starting position is depicted as a red cell on the right.

The robots are networked by the communication protocol described in Section 4. We call PDR the portion of packets sent by a robot that are received by another; PDR<1 means there is packet loss.

The propagation model we implement is based on the Pister-Hack model [9] which is used to obtain the initial Received Signal Strength Indicator (RSSI) between the robots and the orchestrator. This is then translated to a PDR value based on the work done by Municio *et al.* [11]: subtracting a uniform variance of 40 dB from the Friis model equation output and converting RSSI to PDR values by a conversion table that is based on real-world deployments.

6.2 Results

Our aim is to emphasize the importance of considering more realistic communication while designing swarm robotic cooperation algorithms, by demonstrating the impact communication can have on mission time to completion.

In order to adequately demonstrate the impact of PDR, we first run the simulations with various flat PDR rates across any point in the environment from 0.1 to 1 in steps of 0.1. We run the simulations with a swarm of 50 robots. We then run the simulation with the Pister-hack model which generates different PDRs based on distances between robots and the orchestrator. All results are presented with a 95% confidence interval.

Fig. 5 demonstrates the behaviour of the swarm during exploration with various static PDRs. We can see that the behaviour does

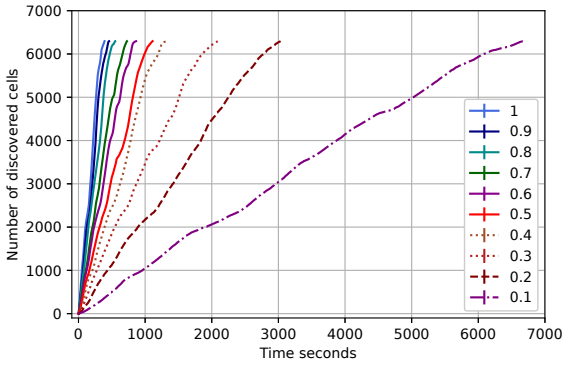

Figure 6: Mapping profiles for different PDRs: the number of cells discovered over time

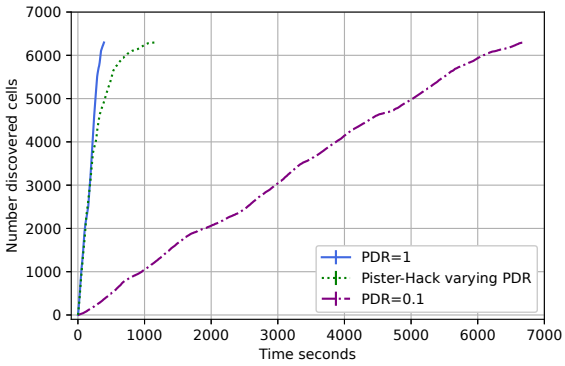

Figure 7: Mapping profile with Pister-hack model.

not vary with packet loss. The paths that seem to be taken more than others are very similar all the way from ideal communication with a PDR of 1 (no packet loss) all the way to a PDR of 0.1 where most packets are lost. This also shows how the modified Atlas algorithm is robust to packet loss, as the overall behaviour and exploration strategy is not affected by packet loss.

However, as clearly seen in Fig. 6, what is significantly impacted by PDR is the exploration time. We can see that as the PDR goes down, the exploration rate goes down with it and the time it takes to complete the mapping increases significantly. In the the floorplan use case we tested, mapping completed in 6.5 min with no packet loss, 19 min with 50% packet loss, 1.85 hours with 90% packet loss. In critical missions such as search and rescue, hazard detection or chemical leakage, this delay in completing the mission could cost lives.

Fig. 7 compares the mapping profile with 100% PDR, 10% PDR, and when using the Pister-hack model. With Pister-Hack, PDR gets lower as the distance between the transmitter and the receiver increases. We can therefore see how initially the mapping profile of Pister-hack resembles that of the case with no packet loss. As time passes, the robots get further and further away from the orchestrator (located at the starting point), and hence, the rate of cells explored per second decreases and the mapping gets slower as communication gets lost.

We conclude that the communication quality, the packet delivery ratio, and the communication protocol drastically impact the performance of the swarm. We also deduce that Atlas 2.0 is robust to packet loss; where the overall behaviour and exploration strategy are not affected by packet loss, neither is the accuracy of the map built.

7 CONCLUSIONS AND FUTURE WORK

This paper introduces Atlas 2.0; an extension of the Atlas algorithm by Abu-Aisheh *et al.*, which we augment to include packet loss tolerance. The result is an exploration and mapping solution that guarantees a mapping completion ratio of 100% even with lossy communication. We infer that Atlas 2.0 is robust to packet loss; where the overall behaviour and exploration strategy are not affected by packet loss, neither is the accuracy of the map built. We demonstrate the need for focusing on communication limitations when designing exploration algorithms by signifying the effect of packet loss on mapping time to completion. We run various simulation scenarios on a discrete-event, continuous time and space open-source simulator that integrates lossy communication models. We show how, the higher the packet loss, the longer the mapping takes to complete. We therefore stress the importance of considering methods to compensate for the delay caused by lossy communication when designing and implementing algorithms for micro-robotic swarm exploration.

This research opens up several avenues for future work. One is including non-ideal headings and speeds, as well as incorporating the context of network connectivity into the logic of our algorithms. We are currently working on experimentally validating our work and are therefore building a swarm of 1,000 robots.

REFERENCES

[1] Razanne Abu-Aisheh, Francesco Bronzino, Myriana Rifai, Brian Kilberg, Kris Pister, and Thomas Watteyne. 2020. Atlas: Exploration and Mapping with a Sparse Swarm of Networked IoT Robots. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, IEEE, Marina Del Rey, CA, USA, 338–342.

[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.

[3] Jacopo Banfi. 2019. Recent Advances in Multirobot Exploration of Communication-Restricted Environments. *Intelligenza Artificiale* 13, 2 (2019), 203–230.

[4] Jacopo Banfi, Alberto Quattrini Li, Nicola Basilico, Ioannis Rekleitis, and Francesco Amigoni. 2016. Asynchronous Multirobot Exploration Under Recurrent Connectivity Constraints. In *International Conference on Robotics and Automation (ICRA)*. IEEE, IEEE, Stockholm, Sweden, 5491–5498.

[5] Jacopo Banfi, Alberto Quattrini Li, Ioannis Rekleitis, Francesco Amigoni, and Nicola Basilico. 2018. Strategies for Coordinated Multirobot Exploration with Recurrent Connectivity Constraints. *Autonomous Robots* 42, 4 (2018), 875–894.

[6] Facundo Benavides, Caroline Ponzoni Carvalho Chanel, Pablo Monzón, and Eduardo Grampín. 2019. An Auto-Adaptive Multi-Objective Strategy for Multi-Robot Exploration of Constrained-Communication Environments. *Applied Sciences* 9, 3 (2019), 573.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107. https://doi.org/10.1109/TSSC.1968.300136

[8] Elizabeth A. Jensen and Maria Gini. 2019. *Distributed Autonomous Robotic Systems*. Springer, Cham, Chapter Effects of Communication Restriction on Online Multi-robot Exploration in Bounded Environments, 469–483.

[9] Hanh-Phuc Le, Mervin John, and Kris Pister. 2009. Energy-Aware Routing in Wireless Sensor Networks with Adaptive Energy-Slope Control. *EE290Q-2 Spring* (2009).

[10] Sabato Manfredi, Enrico Natalizio, Claudio Pascariello, and Nicola Roberto Zema. 2017. A Packet Loss Tolerant Rendezvous Algorithm for Wireless Networked Robot Systems. *Asian Journal of Control* 19, 4 (2017), 1413–1423.

[11] Esteban Municio, Glenn Daneels, Mališa Vučinić, Steven Latré, Jeroen Famaey, Yasuyuki Tanaka, Keoma Brun, Kazushi Muraoka, Xavier Vilajosana, and Thomas Watteyne. 2019. Simulating 6TiSCH Networks. *Transactions on Emerging Telecommunications Technologies* 30, 3 (2019), e3494.

[12] Bradley Woosley, Prithviraj Dasgupta, John G Rogers, and Jeffrey Twigg. 2020. Multi-Robot Information Driven Path Planning Under Communication Constraints. *Autonomous Robots* 44, 5 (2020), 721–737.

[13] Zhi Yan, Luc Fabresse, Jannik Laval, and Noury Bouraqadi. 2017. Building A ROS-Based Testbed for Realistic Multi-Robot Simulation: Taking the Exploration as an Example. *Robotics* 6, 3 (2017), 21.

[14] Tsvetan Zhivkov, Eric Schneider, and Elizabeth I Sklar. 2017. Measuring the Effects of Communication Quality on Multi-Robot Team Performance. In *Annual Conference Towards Autonomous Robotic Systems (TAROS)*. Springer, Guildford, United Kingdom, 408–420.