AFRL-AFOSR-VA-TR-2022-0310

DDDAS Anomaly Detection and Response for Attack-Resilient Multi-Agent Systems

Nicanor Quijano
UNIVERSIDAD DE LOS ANDES
CARRERA 1 18 A 12
BOGOTA, D.C.,
COL

06/27/2022
Final Technical Report

Air Force Research Laboratory
Air Force Office of Scientific Research
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
20220627	Final	START DATE 20181201	END DATE 20211130

4. TITLE AND SUBTITLE
DDDAS Anomaly Detection and Response for Attack-Resilient Multi-Agent Systems

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
	FA9550-19-1-0014	61102F
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Nicanor Quijano

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSIDAD DE LOS ANDES
CARRERA 1 18 A 12
BOGOTA, D.C.
COL

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Office of Scientific Research 875 N. Randolph St. Room 3112 Arlington, VA 22203	AFRL/AFOSR IOS	AFRL-AFOSR-VA-TR-2022-0310

12. DISTRIBUTION/AVAILABILITY STATEMENT
A Distribution Unlimited: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report describes our research on anomaly detection and response for dynamic data driven application systems (DDDAS) in order to build attack-resilient multi-agent systems. The report covers the period between January 2019 and November 2021. Besides the achievements presented in our previous reports, we show some new results regarding the practical implementation of the strategies described, and the final details needed to finish all the activities of the project. Due to the Covid-19 pandemic when the lockdown impeded to work on our laboratories, and we were not allowed to hire graduate research assistants, we asked for an extension that was granted on November 2020. The university opened its laboratories on the second semester of 2021 and at the time we were able to hire four undergraduate research students. Hence, all the activities were developed by these undergraduate students, three graduate students, and the two principal investigators.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR	40

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
DANIEL MONTES	000-0000

Final Report 2022

DDDAS Anomaly Detection and Response for Attack-Resilient Multi-Agent Systems

Luis Burbano (Graduate Research Assistant)
Nicolás Londoño-Cuellar (Undergraduate Research Assistant)
Andres Felipe Manrique-Moreno (Undergraduate Research Assistant)
Juan Pablo Martinez-Piazuelo (Graduate Research Assistant)
Ivan David Perez-Moreno (Undergraduate Research Assistant)
Jose Vicente Vargas-Panesso (Undergraduate Research Assistant)
Nicanor Quijano (CO-PI), Full Professor of Electrical Engineering
Sandra Rueda (CO-PI), Associate Professor of Systems and Computing Engineering
Universidad de los Andes, Bogotá, Colombia

Submitted to:
Southern Office of Aerospace Research and Development
Air Force Office of Scientific Research

February 2022

Summary

This report describes our research on anomaly detection and response for dynamic data driven application systems (DDDAS) in order to build attack-resilient multi-agent systems. The report covers the period between January 2019 and November 2021. Besides the achievements presented in our previous reports, we show some new results regarding the practical implementation of the strategies described, and the final details needed to finish all the activities of the project. Due to the Covid-19 pandemic when the lockdown impeded to work on our laboratories, and we were not allowed to hire graduate research assistants, we asked for an extension that was granted on November 2020. The university opened its laboratories on the second semester of 2021 and at the time we were able to hire four undergraduate research students. Hence, all the activities were developed by these undergraduate students, three graduate students, and the two principal investigators.
This final report is organized as follows:

(i) The first section summarizes the objectives and activities of the project, the progress achieved to date, purchases of required elements, and the list of written papers.

(ii) Section 2 presents some preliminaries that are common to the developed strategies.

(iii) Section 3 presents the developed methodology and experimental results.

(iv) Section 4 proposes a novel discrete-time population dynamics to achieve robot formations.

(v) Section 5 presents the development and further practical implementation of developed strategies for testing.

(vi) Finally, Section 7 presents the final conclusions of the developed work.

1 Introduction

Systems with multiple agents have allowed us to develop different strategies to control large-scale interconnected systems. Compared to systems with a single agent, a system with multiple ones can perform tasks such as surveillance easier and faster. The study of such systems has allowed us to model the animal and human behavior, and design control strategies based on those behaviors. Such strategies extend from algorithms based on bird flocking and bee foraging [1], distributed sensing along networks [2], to synchronization of coupled oscillators [3], among others [4, 5].

Evolutionary game theory has an important inspiration on biological populations and, when properly integrated with physical systems, game theory can optimize the system behavior. The usage of such theory has allowed the design of distributed controllers for different applications like control of water systems [6], or synchronization of isolated microgrids [3]. Most of the proposed solutions to the problem use a continuous-time approach. However, to properly implement the strategies on some systems, discrete-time controllers are required. Even if the continuous time controller is stable, the discretization may become unstable. Therefore, theoretical results to ensure that discrete-time controllers are stable must be developed.

Besides the mentioned discretization problem, controllers and systems face another problem because they are vulnerable to attacks. Malicious agents can modify the system information to damage the users and the physical plant. In multi-agent systems, an attack on one agent can pervade the whole system since it sends corrupted information to the remaining agents. Additionally, attackers can modify the information that an agent sends to its neighbors. Some attacks on real systems have shown the necessity of developing an automatic response to face their effects [7, 8].

The project addresses the aforementioned problems, i.e., control of systems using discrete-time population dynamics and mitigation to attacks on control systems. Therefore, the contributions achieved on this project can be summarized as follows. First, we develop a strategy to detect and mitigate an attack on sensors of one of the system agents. The strategy mitigates effects on an attacked agent and prevent attack propagation to the whole system through the communication network. Second, we design a strategy to optimize convex functions using novel discrete-time population dynamics. We develop theoretical results to ensure system stability. This novel development has allowed us to design a controller to achieve robot formations. Third,

we develop a strategy based on software-defined network (SDN) to mitigate attacks on the communication links. We illustrate that using the SDN gives to cyber-physical systems different capabilities to mitigate attacks in the communication between agents. Finally, we do not only simulate, but also implement some of the aforementioned strategies on a system with multiple differential drive robots to show their efficiency.

1.1 Objectives and activities

Design and implement a formation controller for unmanned ground vehicles (UGVs) capable of mitigating attacks on sensor readings. The work focuses on formations with at least three robots, and limited to integrity and replay attacks.

The objective previously stated can be separated in the following specific objectives:

- Develop a formation controller, for several UGVs, i.e., three to six robots, and at least three geometrical formation distributions.
- Develop a mechanism capable of detecting anomalies (attack/failure) of robot sensors.
- Develop at least one mechanism capable of reconciling measured and estimated values and of computing the required adjustment on control actions to mitigate the effect of anomalies on formations of robots.
- Develop a performance index of the mechanism, to quantify the reduction of the impact of an attack when the mitigation mechanism is added to the formation controller.

To accomplish these objectives, we have stated the following activities on the project proposal:

(i) Design and implementation of the controller for the formation leader.

(ii) Design and implementation of the mechanism to mitigate attacks on the path following task of the formation leader.

(iii) Selection of the set of formations to explore, i.e., definition of robot number and geometrical shape of each formation (at least three scenarios).

(iv) Design and implementation of the controller for the follower robots in each formation for the three robots system.

(v) Development of the mechanism to detect anomalies on the information sent by the leader and/or received by the two followers (that under attack could be different).

(vi) Development of the mechanism to mitigate the effect of attacks on both follower robots in the formation.

(vii) Presentation of partial results in a specialized conference. This publication will include mitigation of attacks on leader and follower robots of one formation with three robots.

(viii) Design and implement the formation controller for the leader and followers in the multi-agent system of six robots.

(ix) Development of the mechanism to detect anomalies on the information sent by the leader and/or received by the followers (that under attack could be different) for the system with six agents.

(x) Expansion of the mitigation mechanism for six agents and different formation shape.

(xi) Definition of a performance index to quantify the difference between the attacked system and the attacked system that includes the mitigation mechanism.

(xii) Publication of final results in a specialized journal.

Robot Name	University/Institution	Odometry	IR Beacon	IR proximity	Ultrasonic/IR distance sensor	Accelerometer	Gyroscope	Bump Sensor	Cliff Detector	Temperature sensor	Camera	Microphone	Other features	Communication	Processor	Retail cost	Parts cost
Khepera IV	K-Team	x		x	x	x	x		x		x	x		802.11 b/g WiFi, Bluetooth 2.0 EDR	Linux core running on a 800MHz ARM Cortex-A8 Processor with C64x Fixed Point DSP core and additional microcontroller for peripherals management	3180	-
Scribbler 3	Parallax	x		x								x	Line sensor		Multicore Propeller P8X32A	179	-
Finch	Finch			x		x				x					Atmega 16/32U	99	-
robomote	USC	x		x									Compass		IC MCU 8K 4MHZ A/D LV 44TQFP	-	150
3pi	Pololu												Line sensor		ATmega328	99	-
CostBots	Berkeley					x									ATmega8L	-	200
e-puck	EPFL	x		x		x					x	x		Bluetooth	dsPic 30F6014A @ 60 MHz (~15 MIPS) 16 bits micocontroller with DSP core	979	-
kilobot	Harvard			x										IR. It can communicate with neighbors up to 7 cm	Atmega328	130	14
r-one	Rice	x	x	x		x	x	x		x				2.4GHz Radio	50MHz ARM Cortex-M3	-	250
e-puck 2	EPFL	x		x	x	x	x				x	x	Extensions to connect Arduino or Raspberry	Bluetooth 2.0, BLE, WiFi	32-bit STM32F407 @ 168 MHz (210 DMIPS), DSP and FPU, DMA	840	-
GRITSbot	Georgia Tech	x		x		x	x							RF transceiver operating at 2.4 GHz	Atmega 168 and Atmega 328	-	50

Figure 1: Comparison of different mobile ground robotic platforms. EPFL is the *École Polytechnique Fédérale de Lausanne* and USC is University of Southern California. Adapted from [9].

1.2 Purchases

To accomplish the aforementioned activities, we have acquired some hardware. Figure 1 shows the main characteristics of some robots that different research groups have built. Although some robots such as Khepera IV present many capabilities, those robots have a high price or they are not commercially available. Therefore, we have selected the e-puck version 2; it has sufficient features to develop the present work and its price allow us to acquire several robots with the available budget.

We have bought a total of six ground robots, a high performance and three medium sized computers. Table 1 shows costs of each component without taxes. Robot prices are different because they were acquired on different dates. Additionally, to implement the robots communication in a distributed fashion, six Raspberry-Pi have been bought, and in order to detect the position of the robots a camera was acquired. As the project counterpart, some resources from the budget of PhD student Luis Francisco Cómbita with the grant Colciencias 727 of 2015 have been used to acquire project supplies, as well as some resources from the budget of PhD student Jorge Alfredo Lopez Jimenez. Colciencias is the Colombian equivalent of the National Science Foundation (NSF).

Table 1: Elements acquired for the project development

Element	Price (USD)	Purchase date
3 e-puck robots	2520	May 2018
3 e-puck robots	2760	November 2018
3 Dell computers	3100	December 2018
1 computer	3530	December 2018
6 Raspberry-Pi	300	October 2019
Camera	50	August 2021

1.3 Schedule Working

The COVID-19 contingency impeded to continue the activities that we were being developed until the beginning of 2020. Since our laboratory at Universidad de los Andes was closed since March 17th 2020 until August of 2021, the experiments on the actual robots were postponed until the second semester of

2021. Furthermore, the univeristy re-opening made it difficult to engage new graduate research assistants, so undergraduate assistants were included. The financial report includes all information related to the researchers hired.

Table 2: Activities plan and development

	YEAR-1				YEAR-2				YEAR-3				
Activity	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Progress
(i)													100%
(ii)													100%
(iii)													100%
(iv)													100%
(v)													100%
(vi)													100%
(vii)													100%
(viii)													100%
(ix)													100%
(x)													100%
(xi)													100%
(xii)													100%

1.4 Publications, Presentations, and Awards

1.4.1 Publications

(i) L. F. Combita, A. A. Cardenas, and N. Quijano, "Mitigating sensor attacks against industrial control systems," IEEE Access, vol. 7, pp. 92 444–92 455, 2019.

(ii) G. Díaz-García, L. Burbano, N. Quijano, and L. F. Giraldo, "Distributed MPC and potential game controller for consensus in multiple differential-drive robots," in Proceedings of the 2019 IEEE 4th Colombian Conference on Automatic Control (CCAC), 2019, pp. 1–6.

(iii) J Martinez-Piazuelo, G Diaz-Garcia, N Quijano, LF Giraldo, "Distributed formation control of mobile robots using discrete-time distributed population dynamics", IFAC-PapersOnLine 53 (2), 3131-3136, 2020.

(iv) A.Murillo, S.Rueda, "Access Control Policies for Network Function Virtualization environments in Industrial Control Systems", Proceedings of the 4th Conference on Cloud and Internet of Things (CIoT) (ISBN 978-1-7281-9541-4), 2020.

(v) JS Gonzalez–Rojas, J Martinez–Piazuelo, N Quijano, "Distributed Assignment with Energy Constraints of a UAV fleet for Resource Distribution", in Proceedings of the 2021 IEEE 5th Colombian Conference on Automatic Control (CCAC), 68-73, 2021.

(vi) J Barreiro-Gomez, I Mas, JI Giribet, P Moreno, C Ocampo-Martínez, R.Sánchez-Peña, N.Quijano, "Distributed data-driven UAV formation control via evolutionary games: Experimental results", Journal of the Franklin Institute 358 (10), 5334-5352, 2021.

(vii) L Burbano, LF Cómbita, N Quijano, S Rueda, "Dynamic Data Integration for Resilience to Sensor Attacks in Multi-Agent Systems", IEEE Access 9, 31236-31245, 2021.

Accepted papers:

(i) J. Martinez-Piazuelo, G. Díaz-García, N. Quijano, L. F. Giraldo, "Discrete-Time Distributed Population Dynamics for Optimization and control", To appear IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022.

Submitted papers:

(i) L.F.Cómbita, N.Quijano, A.Cárdenas, "On the Stability of Cyber-Physical Control Systems with Sensor Multiplicative Attacks", submitted to the IEEE Access, 2021.

1.4.2 Presentations

(i) Sandra Rueda, "Leveraging Software-Defined Networking for Data-Driven Reconfiguration of Control Systems to Survive Attacks", Window on Science Visit (SOARD) to Rome, NY, USA, 2019.

(ii) L.Burbano, L.F.Cómbita, N.Quijano, "Dynamic Data Integration for Resilience to Sensor Attacks in Multi-Agent Systems", Window on Science Visit (SOARD) to Rome, NY, USA, 2019.

(iii) Andrés Felipe Murillo, "Asegurando Sistemas Virtuales de Control Industrial usando Redes Definidas por Software y Virtualización de Funciones de Red.", Foro nacional de seguridad de la información. UniAndes. Bogotá, Colombia, 2019.

(iv) G. Díaz-García, L. Burbano, N. Quijano, and L. F. Giraldo, "Distributed MPC and potential game controller for consensus in multiple differential-drive robots," IEEE 4th Colombian Conference on Automatic Control, Medellín, Colombia, 2019.

(v) J Martinez-Piazuelo, G Diaz-Garcia, N Quijano, LF Giraldo, "Distributed formation control of mobile robots using discrete-time distributed population dynamics", IFAC World Congress, Berlin, Germany, 2020.

(vi) N.Quijano, "Distributed Network Optimization for Control", Engineering Seminar Universidad Pontificia Bolivariana, B/manga, Colombia, April 2021.

(vii) N.Quijano, J.Martinez-Piazuelo, "Distributed Network Optimization for Control", Engineering Seminar Feria de Colegios, Universidad de los Andes, Bogotá, Colombia, May 2021.

(viii) N.Quijano, J.Martinez-Piazuelo, "Distributed Network Optimization for Control", Engineering Seminar Universidad de Nariño, Pasto, Colombia, May 2021.

(ix) N.Quijano, "von Neumann meets the Nexus Problem", Webinar Comité Español de Automática (CEA), Spain, November 2021.

(x) N.Quijano, "von Neumann meets the Nexus Problem", IRI Seminars, Institut de Robòtica I Informàtica Industrial, UPC, Barcelona, Spain, December 2021.

(xi) N.Quijano, "von Neumann meets the Nexus Problem", Winter 2022 Seminar Series, Cyber-Physical Systems Research Center, University of California Santa Cruz, USA, Feb. 2022.

1.4.3 Awards

- Nicanor Quijano received the Experienced Research Award 2021 from the School of Engineering, December 15th, 2021.

2 Preliminaries

2.1 Implementation

We have developed a decentralised communication architecture between robots. On this network, the computation of each robot is performed in a distributed fashion. Some capabilities are still performed on a

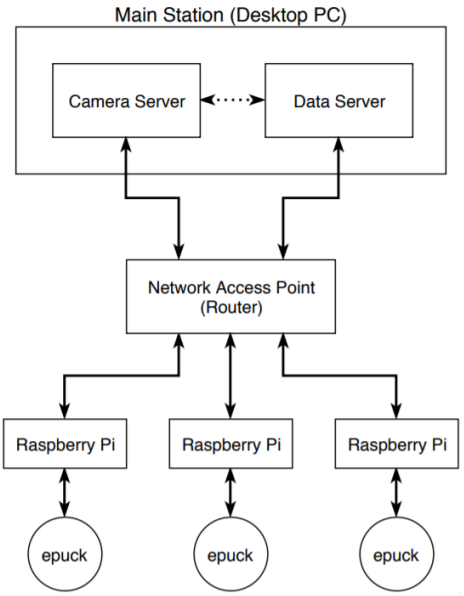


Figure 2: Distributed communication between e-puck robots.

central computer. For example, the camera reading and communication topology definition are performed in the central computer. This is an advantage since we can develop some strategies using techniques such as software defined network (SDN). A diagram that summarizes the new implementation is presented in Figure 2. A Raspberry-pi is connected to each robot to perform the local computations (e.g., robot sensors interpretation). Additionally, each Raspberry is responsible to receive camera reading from the central computer, information from other robots, and to send wheels speeds to the connected robot. All the aforementioned devices are connected via a Wi-Fi through a router.

Two sensors have been used to measure the robots position: the odometry using the motors step counting and a camera. The first sensor is included in the robot motors, while the camera is a sensor on the top of the test platform. The step counting is updated every $0.05\,s$, and the step counting changes 1000 steps per revolution. Camera sensor can measure the robots absolute position, it can take 30 frames per second and a resolution of up to 1920×1080 pixels.

2.2 Graph theory

The communication between robots can be modeled with a weighted directed graph $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k], \mathbf{A}[k])$ with a finite set of vertices $\mathcal{V} = \{1, 2, ..., N\}$, a set of edges $\mathcal{E}[k] \subseteq \mathcal{V} \times \mathcal{V}$ that represents the communication links, and the $N \times N$ adjacency matrix $\mathbf{A}[k] = [a_{ij}[k]]$ whose elements $a_{ij} \neq 0$ if there is an edge from node j to i, and $a_{ij} = 0$ otherwise. The edge (v_i, v_j), denoted as (i, j), means that node j can obtain information from node i. The set of nodes that sends information to the i-th node at time k, called neighbors in a graph, is denoted by $\mathcal{N}_i = \{j \in \mathcal{V} : a_{ij} \neq 0\}$. The graph Laplacian matrix is given by,

$$\mathbf{L} = \mathbf{D} - \mathbf{A},$$

where $\mathbf{D} = diag(\mathbf{A}\mathbf{1}_N)$ is the degree matrix. A directed path is a set of edges that joins a sequence of different vertices. A directed graph is connected if there is a directed path between every pair of nodes. The communication between two nodes i and j is bidirectional if $(i, j) \in \mathcal{V}$ and $(j, i) \in \mathcal{V}$. If the communication between all nodes is bidirectional, then the graph is undirected. In this case, the adjacency matrix is symmetric.

2.3 Continuous time robot model

Differential drive robots are composed by two parallel wheels attached to independent actuators. The left and right actuator can move the wheels at different velocities ω_r, ω_l. The positions x, y, and the angle θ with respect to the x axis can be modeled with the kinematics of a unicycle, which describe the robot as a particle with tangential velocity v and angular velocity ω, i.e.,

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta)\,. \\ \dot{\theta} &= \omega \end{aligned} \tag{1}$$

For the differential drive robot, the linear and angular velocities are given by,

$$\begin{aligned} v &= \frac{r}{2}(\omega_r + \omega_l) \\ \omega &= \frac{r}{L}(\omega_r - \omega_l) \end{aligned},$$

where r is the wheel radius, L is the distance between wheels, ω_r and ω_l are the right and left wheel velocities, respectively.

2.4 Software implementation

We have developed a software implementation of the decentralized control of agents, by simulation and also by real interaction. This system implementation is versatile because it allows the user to define the number of agents, their formation, and their initial position.

2.4.1 Initial parameters

Initially the system allows the user to define variables directly on the code or by asking them though console manners for both physical implementation or by simulation. These initial variables are:

- **The number of agents:** This parameter is associated with information flow through the system as the user define the quantity of the agents he wants to work with, so the Association Matrix change according to this number.
- **The distances among those agents:** Defines the algorithm and the mathematical description of the system which in turn allows different agents to get to a desired formation.
- **The connections:** As we work on a decentralized control system, it is necessary to define the connections among agents, because this parameter defines to whom agents a specific agent can access so it can make its calculus based on their data. The user has to define whether each agent is able to communicate among each other to perform its calculus or not. This means the user has to define the access level each agent has to other agents present state. An agent does not have access to the states of all the other agents, instead, have the information of the state of some agents defined by the user. This parameter replies to the functioning of real big nets, where would be very expensive and inefficient to establish all the possible connections among the agents, or even some small networks in which researchers might be interested in observing dynamics of a small group of agents whose communications are restricted to some specific conditions.
- **Initial positions:** In the event of the simulation, the user defines the initial position of each agent according to his desires. On the other hand, real implementation changes the way these initial conditions are settled, as in this process the camera over the robots is the one in charge of defining the position of each agent using techniques of image processing, and color recognition.

2.4.2 Automatic process

According to the decentralized process, there are three big automatic processes. The first one is access to the state of the agent every iteration. In the simulation, the agent makes the calculus of his position every iteration and assumes this information is correct. On the other hand, in a real application, the agent needs to ask for this information from the central station which has access to the camera and hence knows their real-time position. The second one is the state of the other agents, according to the initial user definition the central station allows having detailed control of what sort of information does an agent has and the information it is able to have.

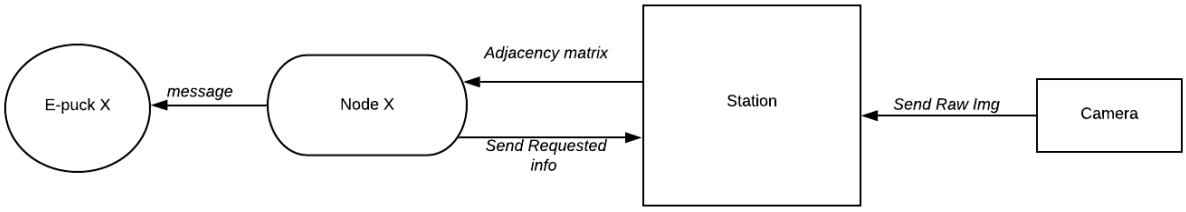


Figure 3: Central station implementation for defining communication network.

Finally the third process is to implement the calculus of the movement that the agent have to do according to the information received. Implementation of such calculus requires the usage of controllers that enables the system to adequately control agents movement through physical space. This was achieved by implementing a PID controller.

Algorithm 1: User interface

Data: Robot list, $\mathcal{R} = \{\}$. Robot distances matrix, **D**.Connections Matrix, **C**. Positions, **P**.
Result: Development of the process of formation
The user defines the number of robots. $\mathcal{R}$ is updated
The user defines the distances among robots. $\mathcal{D}$ is updated
The user defines the connections of each robots. $\mathcal{C}$ is updated
foreach *robot i in* $\mathcal{R}$ **do**
    Receive information of the nodes have access to
    Receive information of the state based on the camera
    Make its own calculations according to these data
    Make a move to achieve the formation desired
    Update position of the robot on P

2.4.3 Image processing and real-time positioning

Implemented algorithm for real-time agent detection based on color is based on HSV format properties that allow to perform this process efficiently. In general, this format contains three channels of information per pixel, the first (H) stands for the hue and it contains relevant information regarding to the color. The second channel (S) stands for saturation and finally the last channel (V) or value that refers to pixel brightness. Different filters were initialized based on expected color values on real agents so that the algorithm goal would be to detect such agents by detecting regions of the image where color values were similar to those defined on the filters.

Firstly, each frame is binarized by thresholding operations based on filter information. In the end, this allows to obtain a mask containing convex elements of the image that passed threshold. Secondly, morphological operations (aperture) are applied on the image so that noise related to frame capture and general mask quality improves. Finally, contours are localized based on previous information and coordinates

of each agent are calculated. Notice that it is equally relevant to know where an agent is as well as its orientation related to a common reference system. That is why agent design should include a way to differentiate agent direction of movement. In this report, two circles of the same color but differently sized were used on each agent so that once circles were localized, the radio of each might be calculated as well as the coordinates of their centers, which means that if it can be possible to guarantee this circles are going to stay still thought out the process, then angles and directions of the agents can be determined easily.

2.5 Robot configuration and operation

Next, to obtain a physical response with real agents, e-puck robots were used on the platform. These robots can be operated via Bluetooth and WiFi. For the purposes of the project where we have several of these e-pucks, the WiFi communication protocol was chosen. This protocol is based on a Client-Server service, where the e-puck works as a server in which the received information is processed and the requested action is executed. Now, the e-pucks come by default with eight types of commands or messages depending on what you want to achieve with the robots, for this case the message used is the -0x09 because this message is used to set the values of the robot actuators, such as motors.

Table 3: Messages Structure

Settings	Motors				LEDs	RGB LEDs				Speaker
Flags	Left	Left MSB	Right LSB	Right MSB	State	LED2	LED4	LED6	LED8	Sound id
(1)	(1)	(1)	(1)	(1)	(1)	(3)	(3)	(3)	(3)	(1)

Likewise, each robot is configured and linked to its respective raspberry pi, which are the ones that communicate directly with the robots and the central computer that acts as a router. In this way, the robots are assigned an IP address with which the connection with their respective Raspberry pi is established. Finally, with the connection established, each Raspberry pi is in charge of telling the associated robot where it should move by sending structured messages such as those mentioned in Table 3.

2.6 Anomalies and attacks detection

In order to simulate the behavior of the agents under a real situation two types of of attacks were defined.

The first one is defined as a communication attack were the communication between two agents is corrupted. On this case, only the information that the agent A sends to the agent B is wrong, leading to two possibilities: i) the attacker predefined a state for the agent A; or ii) the attacker adds a fraction of error to the real state of the agent A, which implies that the agent B receives the real state of the agent A plus an error factor.

The second kind of attack is when a node is attacked, i.e., node-attack that makes that all the information sent by this agent will be wrong. As on the previous attack, the attacker could predefine a state for the agent infected, or he can add an error factor on the state of the agent. All these type of attacks can be seen in Figure 4.

3 Dynamic Data Integration for Resilience to Sensor Attacks in Multi-Agent Systems

3.1 Literature review

The number of works that make automated decisions to mitigate attack impacts have increased in the last years [10]. Regarding attacks on sensors and actuators of ground robots, most of the works focus on detection. These works implement a monitor to verify that the system is properly working. For that purpose, some works use analytical redundancy to verify that the behavior of a system is similar to the behavior generated with a mathematical model. The work in [11] proposes a linear monitor that accumulates the quadratic error in a time window.

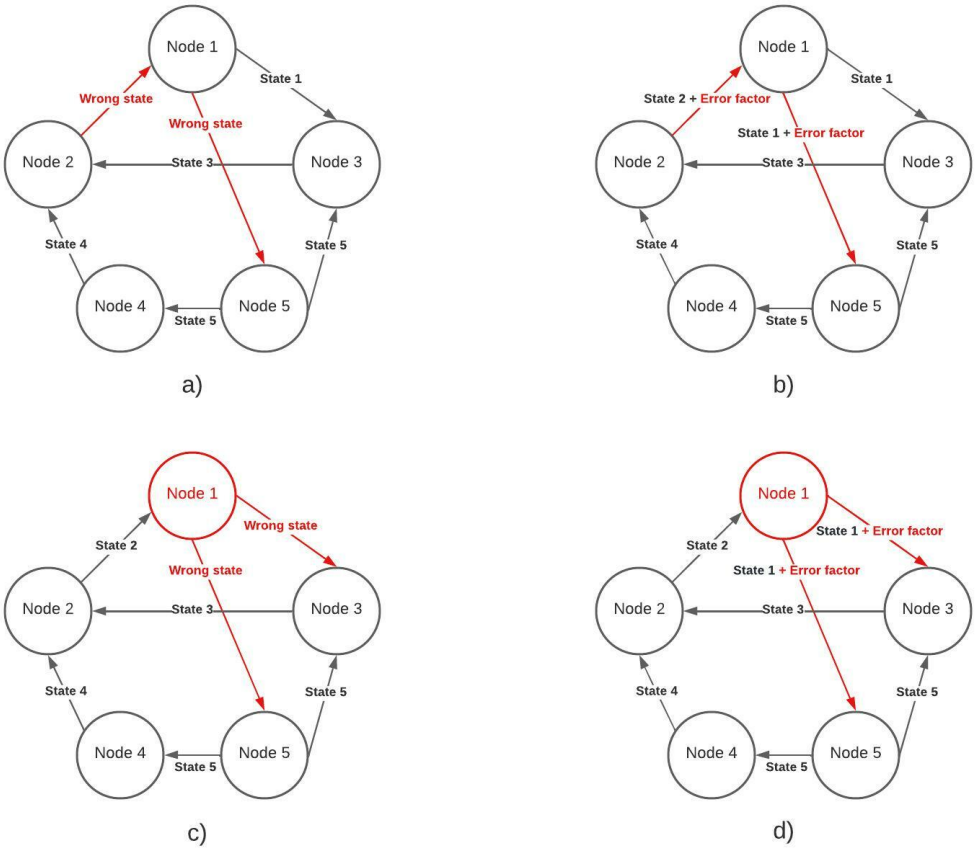


Figure 4: Description of different kinds of attacks. On a) and b) there are shown the two types of communication attacks. a) The attacker defines a wrong state for the communication sent. b) An error factor is added to the real state of the agent. On the other hand, on c) and d) are nodes attacks on node 1. c) A wrong state is sent to all the other nodes. d) An error factor is added to the real state of node 1.

As vehicles present nonlinear dynamics, authors in [12] use a monitor with a nonlinear model. The work in [13] presents a hardware-based redundancy strategy that uses multiple sensors to identify an attack; the authors propose a modified principal component analysis (PCA) to find the envelopes where the system is working without attacks. Then, an attack can be identified if a sensor is outside such envelope.

Although the mentioned works detect various attacks, they do not propose any automated mitigation strategy. Regarding strategies to mitigate attack effects on ground robotic systems, we found works that either focus on single robot systems or multi-robot systems. In the first group, the work in [14] uses a linear model to estimate system output and detect an attack. Once an attack is revealed, the controller is fed with the estimated output. Additionally, the work in [15] proposes a modification of the linear Kalman filter such that the influence of the attacked sensor is decreased. Regarding the works in multi-agent systems, authors in [16] propose a moving target defense (MTD) strategy to decrease the effects of attacks on the communication network. The work shows that randomly changing the communication graph can reduce the deviation produced by the attack. This strategy is finally extended to attacks on system's sensors and controllers. A formalization of the strategy to face attacks is developed in [17]. The strategy presented in [18] mitigates attacks by isolating communication links and nodes in unmanned aerial vehicles (UAV) formations. To detect such attacks, each agent implements an unknown input observer (UIO) bank. Once the attacked agent has been identified, it is removed from the UAV formation [19]. However, in those works the reconfiguration strategy modifies formation geometry; a strategy to mitigate sensor attacks in multi-agent systems, preserving formation geometry, is needed.

3.2 System description

In robotic formations, one of the main objectives is to maintain relative distances between the involved robots. One approach to achieve a particular formation is to use a leader-follower structure. While the leader robot moves in the space tracking a reference without depending on the neighbors position, the follower robots must modify their positions to maintain formation. Therefore, to accomplish the objective, each agent must implement a controller to achieve a position and each robot must share its position with a set of local robots (neighbors in a graph). Such formation and controller can be affected by attacks on the agent This section presents the controller and the threat model. The robot model, controller and threat model are presented in this section. The interconnection of those elements are presented in the blue dotted box in the block diagram in Figure 5, which shows the complete reconfiguration strategy.

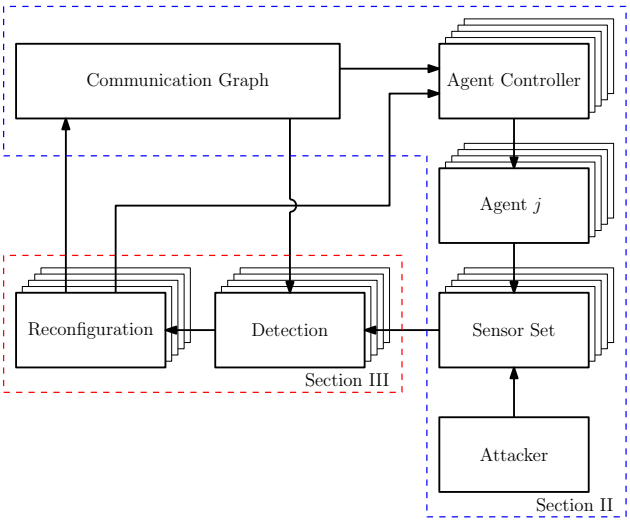


Figure 5: Complete strategy proposed to mitigate attack effects on agent sensors.

3.2.1 Controller

The controller that we have implemented to achieve the system formation is a feedback linearization. This technique linearizes the relationship between the system input and output. Thus, a linear controller can be used to achieve a formation shape. For the differential drive robot, let us define the output as

$$\check{\mathbf{z}} = \begin{bmatrix} x + d_{FL}\cos(\theta) \\ y + d_{FL}\sin(\theta) \end{bmatrix}, \tag{2}$$

with $d_{FL} > 0$. The feedback linearization controller is given by,

$$\mathbf{u} = \frac{1}{d}\begin{bmatrix} d_{FL}\cos\theta & d_{FL}\sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\mathbf{u}_{FL},$$

where $\mathbf{u}_{FL} \in \mathbb{R}^2$ is the vector of the tangential velocities in the direction of the first and second component of $\check{\mathbf{z}}$, respectively. The zero dynamics are stable (see [20] and [21] for details).

We assume that the robot 1, denoted with the lower index 1, is the leader, and also that the follower robots denoted with the lower index i, share information via an unweighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set that represents the N robots, and $\mathcal{E}$ is the set that represents the communication links between robots. The leader robot uses a proportional controller to follow a desired position $\check{\mathbf{z}}_{1,d} \in \mathbb{R}^2$, hence

$$\mathbf{u}_{FL_1} = k_p(\check{\mathbf{z}}_1 - \check{\mathbf{z}}_{1,d}).$$

The i^{th} follower robot has the control law

$$\mathbf{u}_{FL_i} = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\left[\check{\mathbf{z}}_j - \check{\mathbf{z}}_i - \mathbf{d}_{ij}\right], \tag{3}$$

where $\mathbf{d}_{ij} \in \mathbb{R}^2$ is the relative distance between agents i and j and depends on the robot formation geometry. Under conditions without attack, a system using this controller converges if there is a directed path (a set of edges that connect a sequence of different vertices) between the leader and all the agents. Note that we can measure all the system variables, and the output defined in (2) is only used to achieve system formation.

3.2.2 Threat model

We assume that for every measured variable in the system, there are redundant information sources (i.e., different sensors). Let $\mathcal{S}_j = \left\{s_j^{(1)}, ..., s_j^{(n_s)}\right\}$ be the list of n_s redundant sensors for the j agent, whose measurements (possibly compromised) are $\left[\bar{\mathbf{z}}_j^{(1)}, ..., \bar{\mathbf{z}}_j^{(n_s)}\right]$, $\bar{\mathbf{z}}_j^{(i)} \in \mathbb{R}^m$. The purpose of the attacker is to modify sensors data to prevent the controllers to compute the required control action. We assume that the attacker has the actual systems' output and can inject any arbitrary value $\mathbf{a}_j^{(i)} \in \mathbb{R}^m$ at some time to sensor i of agent j. However, we assume that there is at least one sensor on each agent that is not under attack, but the strategy does have information about which sensors are not attacked.

Let $\mathcal{K}_j^{(i)} = \left\{k_{j,f}^{(i)}, ..., k_{j,l}^{(i)}\right\}$ represent the attack duration on the i^{th} sensor of the j^{th} agent, between the first time step $k_{j,f}^{(i)}$ and the last time step $k_{j,l}^{(i)}$. The model of the attack is [22]:

$$\bar{\mathbf{z}}_j^{(i)}[k] = \begin{cases} \mathbf{z}_j^{(i)}[k] & \text{for } k \notin \mathcal{K}_j^{(i)} \\ \mathbf{a}_j^{(i)}[k] & \text{for } k \in \mathcal{K}_j^{(i)} \end{cases}, \tag{4}$$

where $\mathbf{z}_j^{(i)}[k]$ is the actual reading of the i^{th} sensor of j^{th} agent, $\bar{\mathbf{z}}_j^{(i)}[k]$ is the measurement reported by the i^{th} sensor of j^{th} agent, and $\mathbf{a}_j^{(i)}$ is the attack signal affecting the i^{th} sensor of j^{th} agent. The attacker can also attack the pre-processing stage implemented for the sensors (e.g., the position estimation using robot encoders).

The threat model has some additional assumptions. First, as we are focusing on attacks to the sensors, we assume that an attacker only deploys attacks to those devices measurements. By a similar reason, we assume that the integrity of the information sent through the communication links is preserved and each agent trusts the information received from its neighbors. As a consequence, if there is an attack to an agent's sensor, this agent will send false data to its neighbors, potentially affecting the whole system. Finally, we also assume an attacker knows the system model, controller, agent neighbors state, detection strategy parameters, and that attacker can deploy a stealthy attack with that information. We argue that if the strategy can mitigate such a strong attacker, it can also mitigate less intelligent attacks.

3.3 Attack Detection and Mitigation

This section proposes a model-based strategy to detect and mitigate attacks on sensors of cyber-physical systems with multiple agents. The strategy uses redundant sensors to measure the system output and every reading is monitored using analytical redundancy. Such redundancy compares the sensor measurements with the expected ones given by the system model. Using this information, the redundant sensors information is fused to estimate the system output without attack. The overall strategy and the integration of both stages is inside the red box in Figure 5, and a detailed procedure of the strategy is presented in Algorithm 2. In the algorithm, $\mathbf{0}_m$ is a vector of zeros with m rows. In this section, the agent subindex is avoided for notation simplicity. However, all the variables refer to only one agent.

The detection stage is performed with an extended Kalman filter (EKF). The filter estimates the system states integrating the model with the input and sensor measurements, while handling system uncertainties such as noise. Let a discrete time nonlinear system be given by,

$$\begin{aligned} \mathbf{x}[k] &= f_d(\mathbf{x}[k-1], \mathbf{u}[k]) + \mathbf{w}[k] \\ \mathbf{z}[k] &= h_d(\mathbf{x}[k]) + \mathbf{v}[k], \end{aligned}$$

where $f_d : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $h_d : \mathbb{R}^{n_x} \to \mathbb{R}^m$, $\mathbf{w}[k] \in \mathbb{R}^{n_x}$ is a multivariate zero-mean Gaussian process noise with covariance matrix $\mathbf{Q} \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{Q} \succeq 0$, and measurement noise $\mathbf{v}[k] \in \mathbb{R}^m$ with covariance matrix

$\mathbf{R} \in \mathbb{R}^{m\times m},\ \mathbf{R} \succeq 0$. The EKF is iteratively defined as follows. First, the predicted state $\hat{\mathbf{x}}^-[k]$ is calculated evaluating the model on the previous prediction and current input $u[k]$, i.e.,

$$\hat{\mathbf{x}}^-[k] = f_d(\hat{\mathbf{x}}[k-1], \mathbf{u}[k]).$$

Then, the system dynamics are approximated using the Jacobians of the matrix $f_d \in \mathbb{R}^n$ and $h_d \in \mathbb{R}^m$ evaluated at the estimated state $\hat{\mathbf{x}}[k-1] \in \mathbb{R}^n$ and the predicted state $\hat{\mathbf{x}}^-[k] \in \mathbb{R}^n$, i.e.,

$$\mathbf{F}[k] = \left.\frac{\partial f_d}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}[k-1],\mathbf{u}[k]},\ \ \mathbf{H}[k] = \left.\frac{\partial h_d}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}^-[k]}.$$

Using these matrices, the error covariance matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ is predicted by $\mathbf{P}^-[k] = \mathbf{F}[k]\mathbf{P}[k-1]\mathbf{F}^\top[k] + \mathbf{Q}$, where $\hat{\mathbf{x}}[0]$ and $\mathbf{P}[0]$ are the filter initial conditions.

Afterwards, the correction is performed by adding the prediction with the error (i.e., difference between the predicted $\hat{\mathbf{z}}$ and read output $\bar{\mathbf{z}}$) multiplied by the Kalman gain $\mathbf{K}$. This gain depends on the predicted error covariance $\mathbf{P}^-$, measure noise covariance matrix $\mathbf{R}$, and the Jacobian of the observation function, i.e.,

$$\begin{aligned}
\mathbf{e}[k] &= \bar{\mathbf{z}}[k] - h_d(\hat{\mathbf{x}}^-[k]) \\
\mathbf{K}[k] &= \mathbf{P}^-[k]\mathbf{H}^\top[k](\mathbf{H}[k]\mathbf{P}^-[k]\mathbf{H}[k] + \mathbf{R})^{-1} \\
\hat{\mathbf{x}}[k] &= \hat{\mathbf{x}}^-[k] + \mathbf{K}[k]\mathbf{e}[k].
\end{aligned}$$

Finally, the error covariance matrix is updated using

$$\mathbf{P}[k] = (\mathbf{I} - \mathbf{K}[k]\mathbf{H}[k])\,\mathbf{P}^-[k].$$

On a scenario without attacks, changes on the input and output modify the estimation residues $\mathbf{r}[k] = |\bar{\mathbf{z}}[k] - \hat{\mathbf{z}}[k]|$, where $|\cdot|$ is the elementwise absolute value. However, when a sensor information is maliciously modified, the magnitude of the estimation residues increases. To determine such modification on the residues behavior produced by an attack, the non-parametric cumulative sum (CUSUM) is used. Test raises an alarm when the residues magnitude is big or it presents a persistent deviation. CUSUM has shown to detect more attacks and outperform other tests [23]. To mitigate attacks on a system's agent, multiple sensors information is fused. For every sensor, a different EKF and CUSUM is used to monitor the sensor measurement. The CUSUM for the j^{th} measurement of the i^{th} sensor is iteratively defined by,

$$\begin{aligned}
S^{(i,j)}[k] &= \max\{0, S^{(i,j)}[k-1] + r^{(i,j)}[k] - \nu^{(i,j)}\}, \\
S^{(i,j)}[0] &= 0,
\end{aligned}$$

where $\nu^{(i,j)} \in \mathbb{R}_+$ is a CUSUM parameter that prevents the statistic to increase under normal operation, and $r^{(i,j)}$ is the estimation residue for the j^{th} measurement of i^{th} sensor[1]. Every sensor has a discount parameter $\boldsymbol{\nu}^{(i)} = [\nu^{(i,1)}, ..., \nu^{(i,ns)}]$ and it is selected as the minimum value such that, in a scenario without attack,

$$\mathbb{E}[r^{(i,j)}[k] - \nu^{(i,j)}] < 0, \tag{5}$$

where $\mathbb{E}[\cdot]$ is the expected value. This condition ensures that CUSUM is bounded when there are no attacks in sensors, and therefore false alarms due CUSUM unboundedness are avoided. The CUSUM raises an alarm on sensor i when the statistic of the measured variable j exceeds a threshold $S^{(i,j)}[k] > \tau^{(i,j)}$, $\tau^{(i,j)} \in \mathbb{R}_+$. As the parameter $\boldsymbol{\nu}^{(i)}$, every sensor has a threshold $\boldsymbol{\tau}^{(i)} = [\tau^{(i,1)}, ..., \tau^{(i,ns)}]^\top$. These parameters are determined based on a false alarm rate: the bigger the threshold is, the lower the false rate is but detection time increases. In this work, we tune the CUSUM parameters as presented in [22].

The previous paragraphs present a strategy to detect modifications of sensor data. However, we also need an automated response to maintain the system in a safe state and meet the control system objective. The remaining of this subsection shows the proposed mitigation strategy integrated with the detection process.

In order to mitigate the attack effects on the robotic system, a different EKF monitors every reading of each sensor to obtain the residues. Using that information, redundant sensors information is fused. A

[1] $\mathbb{R}_+$ represents the strictly positive quadrant of $\mathbb{R}$

Algorithm 2: Attack detection and mitigation strategy

Data: Sensor list, $\mathcal{S} = \{s^{(1)}, ..., s^{(n_s)}\}$. Sensor measurement, $\bar{\mathbf{z}}^{(i)}[k]$. Sensor error covariance matrix, $\mathbf{P}^{(i)}[k-1]$. Process noise covariance Matrix, $\mathbf{Q}^{(i)}$. Measurement noise covariance matrix, $\mathbf{R}^{(i)}$. Sensor CUSUM, $\mathbf{S}^{(i)}[k-1]$. CUSUM thresholds, $\boldsymbol{\tau}^{(i)}$. CUSUM discount parameter, $\boldsymbol{\nu}^{(i)}$. Sensor state estimation, $\hat{\mathbf{x}}^{(i)}[k-1]$. The input calculated for every sensor, $\mathbf{u}^{(i)}[k]$.

Result: Trust value of output $\mathbf{z}[k]$, $\tilde{\mathbf{z}}[k]$

$\Sigma_\alpha \leftarrow \mathbf{0}_m$;

foreach *sensor i in $\mathcal{S}$* **do**

- $\hat{\mathbf{x}}^{(i)}[k] \leftarrow EKF(\hat{\mathbf{x}}^{(i)}[k-1], \mathbf{u}^{(i)}[k], \mathbf{P}^{(i)}[k-1], \mathbf{Q}^{(i)}, \mathbf{R}^{(i)})$;
- $\mathbf{r}^{(i)}[k] \leftarrow |\bar{\mathbf{z}}^{(i)}[k] - h(\hat{\mathbf{x}}^{(i)}[k])|$;
- **foreach** *measured variable j* **do**
  - **if** $S^{(i,j)}[k-1] > \tau^{(i,j)}$ **then**
    - $S^{(i,j)}[k] \leftarrow 0$;
    - Raise an alarm on sensor i;
  - **else**
    - $S^{(i,j)}[k] \leftarrow \max(0; S^{(i,j)}[k-1] + r^{(i,j)}[k] - \nu^{(i,j)})$;
  - **if** *an alarm has not been raised in sensor i* **then**
    - $\alpha^{(i,j)}[k] \leftarrow 1 - S^{(i,j)}[k]/\tau^{(i,j)}$;
  - **else**
    - $\alpha^{(i,j)}[k] \leftarrow 0$;
  - $\Sigma_\alpha^j \leftarrow \Sigma_\alpha^j + \alpha^{(i,j)}$;

$\tilde{\mathbf{z}}[k] \leftarrow \mathbf{0}_m$;

foreach *sensor i in $\mathcal{S}$* **do**

- **foreach** *measured variable j* **do**
  - **if** $\Sigma_\alpha^j > 0$ **then**
    - $b^{(i,j)} \leftarrow \alpha^{(i,j)}/(\Sigma_\alpha^j)$;
  - **else**
    - $b^{(i,j)} \leftarrow 1/n_s$;
  - $\tilde{z}^{(j)}[k] \leftarrow \tilde{z}^{(j)}[k] + b^{(i,j)}\bar{z}^{(i,j)}[k]$;

confidence value is computed in each iteration for every sensor i and every measured variable j, which is defined by,

$$\alpha^{(i,j)}[k] = 1 - \frac{S^{(i,j)}[k]}{\tau^{(i,j)}},$$

where $\alpha^{(i,j)} = 0$, if an alarm has been raised in the j^{th} measured variable of i^{th} sensor. Note that confidence value is near to zero when CUSUM is near to the threshold, i.e., when there is a difference between the expected and actual behavior. In contrast, the confidence is near to one when the CUSUM does not show a difference between the actual and estimated measurements.

Finally, the sensors information is fused based on a convex combination, i.e.,

$$\tilde{z}^{(j)}[k] = \frac{1}{\sum_{i\in\mathcal{S}} \alpha^{(i,j)}[k]} \sum_{i\in\mathcal{S}} \alpha^{(i,j)}[k]\bar{z}^{(i,j)}[k], \tag{6}$$

where $\mathcal{S}$ is the sensor list and $\bar{z}^{(i,j)}$ is the reading from the i^{th} with the j^{th} variable, and $\tilde{z}^{(j)}$ is the trust value of the j^{th} measured variable. The calculated trust value $\tilde{\mathbf{z}}$ is then used to compute the control action and it is also sent to the communication network to prevent attack propagation.

Finally, Figure 6 presents the DDDAS loop of the designed strategy for the ground robots. In step 1, multiple sensors data (e.g., data from a camera or odometry) are acquired. Later, in step 2, data is fused with a nonlinear model through the EKF and the residues are computed using the estimated outputs.

Attacks are detected using the CUSUM as a classifier, and the confidence values are calculated in step 3. This information, which comes from the fusion of sensor data and a model, is used to take actions on the sensors and fuse the real data in step 4. Finally, the control action is computed using $\tilde{\mathbf{z}}[k]$ with the feedback linearization and it is sent to the robot and its neighbors.

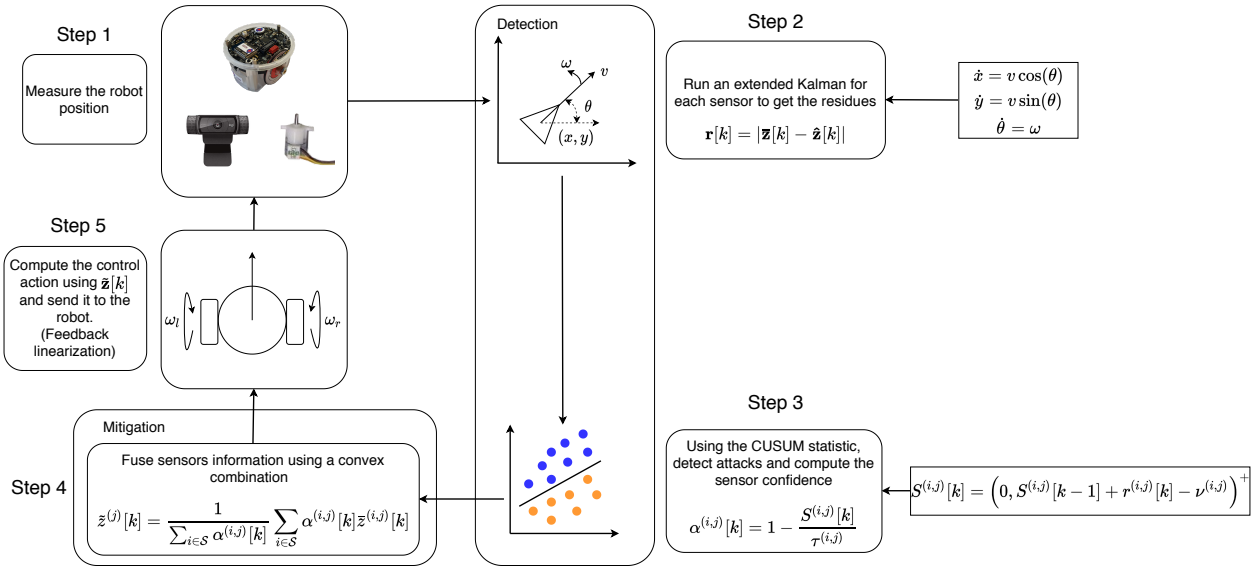


Figure 6: DDDAS loop of the designed strategy.

3.4 Results

This section presents the implementation of some attacks to show the strategy usefulness. To apply the strategy, the discount parameter is $\boldsymbol{\nu}^{(i)} \in \mathbb{R}^3_+$ and threshold $\boldsymbol{\tau}^{(i)} \in \mathbb{R}^3_+$. We will denote the measurement of x, y, θ from sensor i by $\bar{z}^{(i,1)}$, $\bar{z}^{(i,2)}$, $\bar{z}^{(i,3)}$, respectively. The discount parameter of the i^{th} sensor will also be denoted by $\nu^{(i,1)}$, $\nu^{(i,2)}$, $\nu^{(i,3)}$ for the measurement of x, y, θ, respectively. Additionally, as we can measure all the system variables, the output function $h_d(\mathbf{x})$ for the EKF is,

$$\mathbf{z}[k] = h_d(\mathbf{x}[k]) = \mathbf{C} \begin{bmatrix} x[k] & y[k] & \theta[k] \end{bmatrix}^\top,$$

where $\mathbf{C}$ is the 3×3 identity. We also use a discrete-time version of differential-drive robot model (1), i.e.,

$$\begin{aligned} x[k+1] &= x[k] + v[k]\cos\left(\theta[k] + \frac{\omega[k]}{2}T_s\right)T_s \\ y[k+1] &= y[k] + v[k]\sin\left(\theta[k] + \frac{\omega[k]}{2}T_s\right)T_s, \\ \theta[k+1] &= \theta[k] + \omega[k]T_s \end{aligned}$$

where T_s is the sample time.

Note that the integration of the model presented in (1) with the controller and the mitigation strategy can be modeled as a hybrid system. A hybrid system $\mathcal{H}$ can be described with four elements: the flow set $C_\mathcal{H}$, flow map $f_\mathcal{H}$, jump set $D_\mathcal{H}$ and jump map $g_\mathcal{H}$. The jump and flow set are the points where the system flows or jumps, respectively. The behavior of the system when it flows or jumps is described by the flow map and the jump map. Then, to model the mitigation strategy as a hybrid system, we add an additional state β which is a timer. This timer is initialized at 0 and constantly grows at a rate of one, i.e., $\dot{\beta} = 1$, when $\beta \in [0, T_s]$. When the timer is less than the sampling time $\beta \leq T_s$, the system is in the flow set and the continuous states change. That is, the states of the robot are updated according to equation (1). Once the timer increases to $\beta \geq T_s$ the system is in the jump set and the discrete states change. That is, the states

of the robot remain constant while the controller, states estimations, CUSUM, confidence values and trust values are updated. Additionally, the timer is reset to zero $\beta = 0$ and the system arrives again to the flow set [24, 25].

3.4.1 Experimental setup

The strategy has been implemented in a system with 6 e-puck version 2 robots [26]. The robot has a maximum velocity of 15.4 cm/s, radius $r = 2.05$ cm, and distance between wheels $L = 5$ cm. Additionally, it has two stepper motors, a Wi-Fi and bluetooth module, and multiple sensors as the inertial measurement unit.

To measure the robot states, odometry with the motor step counting has been implemented. Every wheel revolution represents 1000 steps and the robot sends its information every 0.05 s. Additionally, to perform the redundancy strategy, a camera is used to measure the robot angle and position. The camera can take up to 30 frames per second at 1080p. However, with such resolution the image recognition is expensive. Therefore, the image size has been changed to 600×450 pixels and 25 frames per second. This results in a resolution of 4 pixels per centimeter. Therefore, to ensure that the sensors are updated at each step, the discretization time has been set to 0.1 s. A central processor runs image recognition, odometry, robot communication, and attack detection.

3.4.2 CUSUM parameters

A proper parameter selection ensures a fast attack detection and low false alarm rate. In this work, we use the procedure presented in [22] to tune the CUSUM parameters. To guarantee the CUSUM boundness condition of (5), 10000 simulations for fifty seconds without attack have been performed to approximate the expected value. Every 1000 simulations, the maximum reference change rate is incremented 1 cm/s until a maximum velocity of 10 cm/s. For every simulation, the residues are calculated for the measured variable (i.e., x, y, and θ) and the mean is calculated to estimate the expected value of the residues. Therefore, the parameters found for both sensors for measurement in x is $\nu^{(i,1)} = 0.0023$, for measurement in y is $\nu^{(i,2)} = 0.0024$, and for measurement of θ is $\nu^{(i,3)} = 0.1103$ for sensors $i = \{1, 2\}$.

Finally, the CUSUM is calculated for every simulation with different threshold values to find the number of false alarms. High thresholds for x, y and θ have been selected to avoid a large false alarm rate, i.e, $\tau^{(i,1)} = 0.8$, $\tau^{(i,2)} = 0.8$, $\tau^{(i,3)} = 4$, for sensors $i = \{1, 2\}$.

3.4.3 Numerical Results

A six-robots system has been used to test the developed strategy. To implement the formation controller and show that the proposed strategy can work with different graphs, two communication topologies are used. The first communication graph, graph 1, used in this paper is defined by $\mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{E}_1\}$, where $\mathcal{V}_1 = \{1, ..., 6\}$ and $\mathcal{E}_1 = \{(1,2), (1,6), (2,3), (3,2), (3,4), (4,3), (4,5), (5,4), (5,6), (6,5)\}$. The second graph, graph 2, is a directed graph that connects the leader and the follower robots with a directed path. This graph is defined by $\mathcal{G}_2 = \{\mathcal{V}_2, \mathcal{E}_2\}$, where $\mathcal{V}_2 = \{1, ..., 6\}$ and $\mathcal{E}_2 = \{(1,2), (2,3), (3,4), (4,5), (5,6)\}$. Figure 7 shows a photo of the system, graph 1 (represented with red arrows) and graph 2 (represented with blue arrows). Note that both graphs have a directed path between the leader and the followers to ensure that robots can achieve the formation when there are no attacks. Additionally, Figure 8 presents behavior for simulation and implementation of the system with a triangular formation and communications defined by graph 1.

The first attack is deployed on the leader robot odometry. Specifically, between 3 and 12 sec., the right motor step counting is stopped. The second attack is deployed on the camera sensor, adding a sigmoidal signal to the measurement of position x of the leader robot:

$$\bar{z}_1^{(2,1)}[k] = z_1^{(2,1)}[k] + \frac{0.2}{1 + e^{-3(0.1k-3)}}, \mathcal{K}_1^{(2)} = \{0, 1, ...\}.$$

It is important to clarify that we have also run two simulations deploying the every attack in robots 1 (leader), 2, 4, and 6 simultaneously. In contrast, only the second attack has been deployed in multiple robots

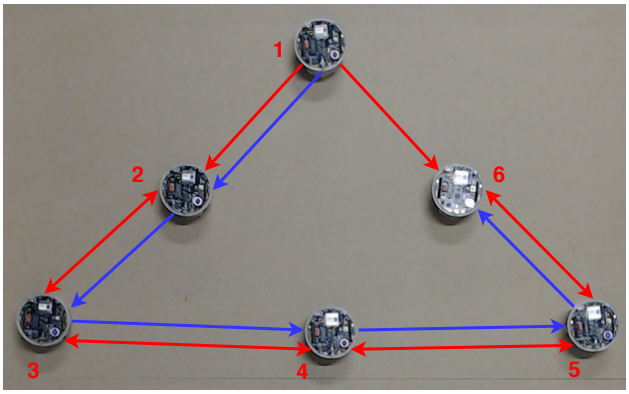


Figure 7: Aerial photography of e-puck version 2 robots formation. The red lines are a graphical representation of the communications graph. The red arrows indicate the flow direction of the information exchange in the graph $\mathcal{G}_1$, and the blue arrows represent the information exchange in the graph $\mathcal{G}_2$.

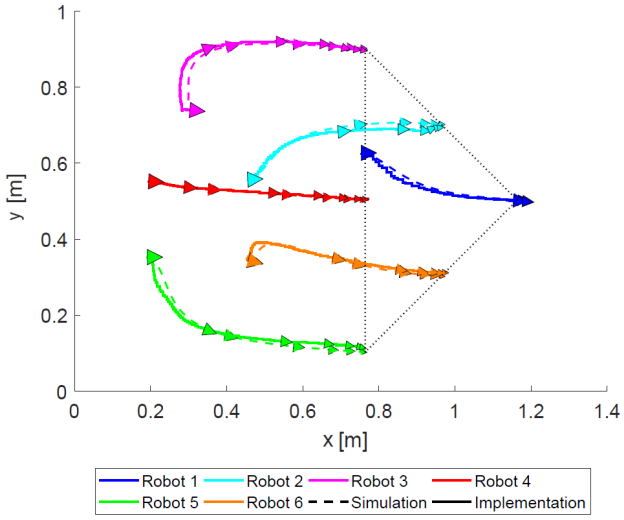


Figure 8: Trajectories without attack on the experimental setup (continuous lines) and simulation (dotted lines), using the graph 1.

in the physical system due to safety reasons. Additionally, a stealthy attack has been deployed on the camera sensor. For a detector that uses the CUSUM statistic, an optimal attack is given by [23],

$$\bar{z}_1^{(2,1)}[k] = \hat{z}_1^{(2,1)}[k] + (\tau^{(2,1)} + \nu^{(2,1)} - S_1^{(2,1)}[k]),\ \mathcal{K}_1^{(2)} = \{20, 21, ...\}.$$

This attack implies that the attacker knows not only the current state, but also the detection strategy and model parameters, i.e., the functions f_d, h_d, robot parameters r, L, CUSUM parameters $\boldsymbol{\tau}$, $\boldsymbol{\nu}$ and EKF covariance matrices $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{P}$. Although this powerful attacker could be unrealistic, we deploy this attack to evaluate strategy results[2].

To quantify the attack mitigation strategy efficiency, a key performance index (KPI) is defined. Let $\check{\mathbf{z}}_i[k]$ be the i^{th} robot output at instant k, and $\check{\mathbf{z}}_{i,d}[k]$ be the i^{th} robot desired output when the formation is

[2]Some videos of the presented attacks can be found at: youtu.be/B0Vy33MbH4Q

Table 4: Normalized key performance index for the different experiments

Graph 1				
	Implementation		Simulation	
Experiment	No reconfiguration	Reconfiguration	No reconfiguration	Reconfiguration
AO* on leader robot	1.46	1.02	1.42	1.06
AC* on leader robot	1.10	1.02	1.13	1.01
AO* on multiple robots	-	1.03	3.24	1.06
AC* on multiple robots	1.70	1.00	1.66	1.05
SA* on leader robot	1.39	1.08	1.30	1.00
Graph 2				
	Implementation		Simulation	
Experiment	No reconfiguration	Reconfiguration	No reconfiguration	Reconfiguration
AO* on leader robot	1.69	1.03	1.59	1.04
AC* on leader robot	1.12	1.00	1.21	1.07
AO* on multiple robots	-	1.00	4.30	1.07
AC* on multiple robots	1.76	1.00	1.80	1.01
SA* on leader robot	1.72	1.00	1.58	1.00

*AC: Attack on camera. AO: Attack on odometry. SA: stealthy attack.

achieved. The KPI is defined as:

$$KPI = \sum_{i=1}^{N} \sum_{k=0}^{K} \|\check{\mathbf{z}}_i[k] - \check{\mathbf{z}}_{i,d}\|_2^2 T_s.$$

To compare the different scenarios (i.e., without an attack, and attacked with and without reconfiguration), the KPI is normalized by the result obtained in the non-attack scenario. Table 4 shows KPIs of the aforementioned experiments and simulations.

Some KPIs constantly increase due to the attack characteristics. The KPI in the table was taken at 30 s when all scenarios have reached their steady-state. At this instant, the KPIs of scenarios with attack and reconfiguration have converged on all performed experiments.

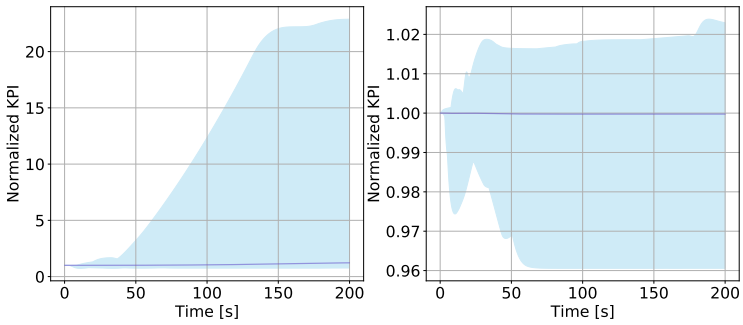
Finally, we have run 1000 simulations, changing initial conditions, time of attacks,attacked agents and communication topology (i.e., selecting either $\mathcal{G}_1$ or $\mathcal{G}_2$). We have selected experiment conditions such that one sensor of every agent may have been attacked at different times during the simulation. We have also selected an intelligent attacker that deploys a stealthy attack. For these simulations we present the normalized KPI in function of time and not only the final value. Figure 9 shows the average of the normalized KPI together with the maximum and minimum KPI obtained in the whole simulation set, for the attack scenarios with and without the presented mitigation strategy. Note that, in order to obtain the normalized KPI, we require the simulation of the scenario without an attack.

3.5 Discussion

The attack on the leader robot odometry has an important effect on the system behavior. At the second 30, the KPI of the experimental setup is greater than the scenario without attack when the graph 1 is used. The KPI shows that performance of the system is similar before and after deploying the mitigation strategy. When an attack starts, it disturbs the system and the KPI grows.

When this attack is deployed on multiple agents, the effects are amplified but the reconfiguration strategy can detect the attack on agents and reduce its effects. Similarly, in our simulations the strategy mitigates attack effects for both scenarios (i.e., attack on leader and simultaneous robots).

The bias attack on the camera does not allow the system to reach the desired formation. Although this attack presents smaller effects than the attack on odometry, the KPI shows that the attack on the leader robot negatively affects system performance. When the attack is deployed on multiple robots, the system performance without the reconfiguration strategy is even worse. In contrast, the system with the



a) b)

Figure 9: Average (blue line), maximum and minimum (blue area plot) normalized key performance index when a stealthy attack is deployed, randomly changing initial conditions, communication graph, attack instants and attacked agents. The KPI is presented when a) mitigation strategy is not implemented, and b) implementing the strategy.

reconfiguration strategy can achieve the expected formation, although the normalized KPI is greater than one. In other words, the system achieves the formation but the attack disturbs the system.

Note that the KPI of scenarios with attacks on multiple agents remain near one, i.e., the system has a similar performance to the scenario without attack. This shows that the reconfiguration strategy can handle attacks on various agents, without affecting the overall system performance. The strategy prevents every agent from misbehaving due to false information from one of its sensors or due to false information sent by a neighbor.

When the graph 2 is used, the attacks effects increase compared to the experiments using the graph 1. In the graph 1, each agent communicates with two agents. Then, when an agent is attacked, its neighbor makes decisions using the information of an attacked and a not attacked robot. In contrast, with the graph 2, all agents receive information from only one agent, which possibly is under attack. Then, with the graph 2, agents could make decisions using only false information even if just one robot is under attack. Nevertheless, with the reconfiguration strategy, the system can achieve the formation when attacks have been deployed on the system. Thus, we conclude that graph topology does not affect the reconfiguration strategy capacity to mitigate attack effects.

As expected, a stealthy attack cannot be detected by the implemented monitor and the CUSUM statistic remains below the threshold. The simulation of multiple scenarios shows that the stealthy attack can produce an important deviation compared to the non-attack scenario. However, when the reconfiguration strategy is implemented, the average, together with the maximum and minimum KPI is near one (i.e., near to the performance of the non-attack scenario). A similar result is obtained when the stealthy attack is deployed in the physical system. This shows that the strategy can mitigate a powerful attacker that modifies an attack to remain undetected, both in simulations and empirical experiments.

Remember that the parameter $\boldsymbol{\nu}$ is selected greater than the residues mean to prevent false alarms produced by the CUSUM unboundedness. However, if the difference between the residues mean and the parameter $\boldsymbol{\nu}$ is too large, the CUSUM would slowly increase and the time spent to detect an attack would also increase. Moreover, small changes in the residues distribution produced by some attacks would not change the CUSUM statistic, which could remain near zero. Therefore, changes in the parameter $\boldsymbol{\nu}$ change the classifier sensitivity. When the parameter $\boldsymbol{\nu}$ is incremented, the detector becomes less sensible to attacks. As a consequence, the system disturbance could become more notorious or the reconfiguration could not be properly made because the controller would be computed with the average between the not attacked and attacked sensors. Thus, the estimated output will be close to the actual output if the number of not attacked sensors is larger than the attacked ones. Similarly, if the parameter $\boldsymbol{\tau}$ is increased, some attacks could become undetectable or the time to detect could increase. However, when the threshold decreases, the number of false alarms could increase and, as a consequence, sensors could remain unused even if there are no attacks.

The simulation and implementation results show that the strategy can reduce attack effects on the system. Those attacks can have similar characteristics to a fault. Additionally, the strategy can identify the attacked

sensor. However, it cannot differentiate between attacks on the sensors or actuators. Therefore, as the strategy only takes actions on the sensors, an attack (or fault) on agent actuators cannot be mitigated. Nevertheless, an alarm would be raised for all sensors and the attack would be detected. In that scenario, as there is not enough information to determine the attack, the trust value is calculated as the average among sensors. The same action would be taken if all sensors are attacked.

Finally, although the simulation and implementation results are similar, there are some differences. Such differences may be explained by three main reasons. First, in the experimental setup, the attacks could be deployed in different time instants by some fraction of seconds. Second, in simulation, both sensors are assumed to be equal in the absence of attacks. However, in the implementation there is an error between measurements obtained from the camera and odometry. Third, the actuator dynamics have not been considered in the simulation. This results in a simpler strategy that requires less physical parameters (e.g., motor inductances and resistances), and less mathematical operations to monitor the sensors because there are fewer state variables to estimate. Nonetheless, the proposed strategy has shown to decrease the effects of attacks applied to real robots using a simplified model.

4 Distributed Formation Control of Mobile Robots Using Discrete-Time Distributed Population Dynamics

4.1 Literature Review

In this part, we focus on the distributed formation control of multiple mobile robots under a leader-follower scheme. More precisely, a leader robot moves through a pre-defined trajectory and some follower robots have to follow the leader while maintaining a desired geometric formation. Moreover, the problem is hardened by the fact that not all followers have information about the leader. Typically, this problem has been solved using the consensus algorithm of [27], but, some recent researches have solved this problem using distributed nonlinear model predictive control [28]; designing distributed estimators for the leader's state [29]; and applying game-theoretical continuous-time distributed population dynamics [30]. As we show in this section, the formation control problem can be formulated as a distributed optimization problem whose solution leads to the achievement of a desired collective behavior among the robots. Following the approach of [30], in this work we apply population dynamics to solve the distributed formation control problem. In contrast with the previous work, however, we propose a novel class of dynamics, formulated in discrete-time, and we provide sufficient conditions to guarantee the asymptotic stability of our method in practical implementations where computations are necessarily discrete. Furthermore, we apply our proposed method to the real multi-robot platform described above, and we compare it against the classical consensus algorithm of [27], and against a game-theoretical distributed optimization method proposed by [31]. Our approach not only outperforms the other game-theoretical method, but also achieves a performance comparable to the consensus algorithm. Moreover, our method displays some invariance properties that make it suitable for other types of distributed control applications including distributed resource allocation [32]. In summary, we propose a novel class of discrete-time distributed population dynamics as a game-theoretical method for decentralized control, and depicts the application of the method to a real multi-robot platform.

4.2 Problem Statement And Control Architecture

In this section we study the distributed formation control of multiple differential-drive robots that move over $\mathbb{R}^2$ under a leader-follower scheme. For such, we consider a set of n robots (one leader and $n-1$ followers) and define the following notations: $\mathcal{V} = \{1, 2, \cdots, n\}$ is the entire set of robots, $\ell = 1$ is the index of the leader robot, and $\mathcal{F} = \{2, 3, \cdots, n\}$ is the set of follower robots. As robots can move over $\mathbb{R}^2$, the location of the i-th robot at time k is given by the tuple $(x_i[k], y_i[k])$ where $x_i[k], y_i[k] \in \mathbb{R}$ for all $i \in \mathcal{V}$ and all k. In this work, we assume that the leader robot follows a pre-defined trajectory within a rectangle of $\mathbb{R}^2$. Without loss of generality, such rectangle is defined as

$$\mathcal{X} = \left\{(x, y) \in \mathbb{R}^2_+ \,:\, x < \beta^x,\, y < \beta^y\right\}, \tag{7}$$

where x and y denote the coordinate dimensions of $\mathbb{R}^2$; $\mathbb{R}^2_+$ denotes the strictly positive quadrant of $\mathbb{R}^2$; and β^x and β^y are positive scalars that define the width and height of the rectangular region $\mathcal{X}$, respectively.

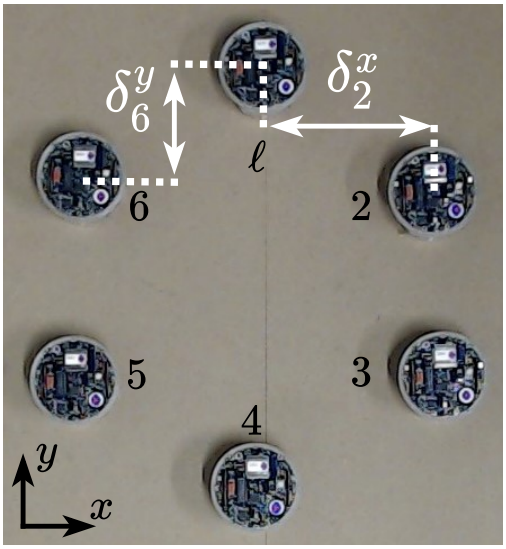


Figure 10: Hexagonal formation for a set of $n = 6$ robots.

On the other hand, the goal of the follower robots is to follow the leader while maintaining a given geometric formation within $\mathcal{X}$. In this work, we assume that robots can communicate with each other through bidirectional communication channels. In general, the interaction between robots can be modeled as an undirected time-varying graph $\mathcal{G}[k] = (\mathcal{V}, \mathcal{E}[k], \mathbf{A}[k])$, where the set of robots $\mathcal{V}$ is the set of nodes; $\mathcal{E}[k]$ is the set of communication links between the robots at time k; and $\mathbf{A}[k] = [a_{ij}[k]]$ is an $n \times n$ adjancency matrix whose elements are $a_{ij}[k] = a_{ji}[k] = 1$ if the i-th robot communicates with the j-th robot at time k, and $a_{ij}[k] = a_{ji}[k] = 0$ otherwise. Moreover, we define $\mathcal{N}_i[k] = \{j \in \mathcal{V} \setminus \{i\} : a_{ij}[k] = 1\}$ as the set of robots that communicate with the i-th robot at time k. The communication between robots is used to obtain the x and y coordinates of other robots, and, in consequence, we assume that at time k the robot $i \in \mathcal{V}$ sends the coordinates $(x_i[k], y_i[k])$ to all robots $j \in \mathcal{N}_i[k]$, and receives the coordinates $(x_j[k], y_j[k])$ of each robot $j \in \mathcal{N}_i[k]$. Whether two robots can communicate might depend on factors like the spatial distance between the robots (as in [30]), or the topology of an available cloud-based communication network (as in [33]). To keep our framework general, in this work we develop our theoretical analyses for arbitrary connected and undirected time-varying graph topologies.

Assumption 4.1. *The undirected communication graph $\mathcal{G}[k]$ is connected for all k.*

To achieve a geometric formation, each follower robot is given a reference displacement vector that defines its required position relative to the leader. Such reference displacement vector is given by $\boldsymbol{\delta}_i = \begin{bmatrix} \delta_i^x, & \delta_i^y \end{bmatrix}^\top$, for all $i \in \mathcal{F}$, where δ_i^x and δ_i^y are the reference displacements for the i-th follower with respect to the leader's x and y coordinates, respectively. For instance, Fig. 10 shows one example formation for a set of $n = 6$ robots and depicts some reference displacements. In this work, we assume that the reference displacement vectors of all the follower robots satisfy the following assumption.

Assumption 4.2. *The required formation lies within $\mathcal{X}$ for all k.*

It is worth noting that depending on the graph topology, not all followers might have communication with the leader. Thus, the presented formation problem is a distributed control task under partial information. To solve such kind of problem, we can break the objective into two tasks. The first one is to determine, distributedly over $\mathcal{G}[k]$, the set-point coordinates in $\mathcal{X}$ that each follower has to reach so that certain formation is achieved. This task can be formulated as a set of time-varying optimization problems that each follower robot has to solve. More precisely, such optimization problems are given by

$$\min_{r_i^d[k]} \frac{\left(c_\ell^d[k] + \delta_i^d[k] - r_i^d[k]\right)^2}{2}, \forall d \in \mathcal{D}, \forall i \in \mathcal{F}, \forall k, \tag{8}$$

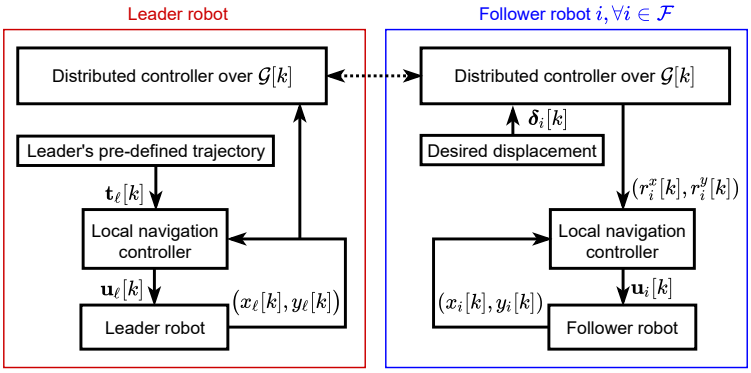


Figure 11: Control architecture for the distributed formation problem.

where the set $\mathcal{D} = \{x, y\}$ contains the dimensions of the space where the robots move; the term $c_\ell^d[k]$ represents the value of the d-th dimension of the leader's location at time k, i.e., $c_\ell^x[k] = x_\ell[k]$ and $c_\ell^y[k] = y_\ell[k]$; the term $\delta_i^d[k]$ is the reference displacement over the d-th dimension for the i-th follower at time k; and $r_i^d[k]$ is the d-th component of the set-point coordinate for the i-th follower at time k, i.e., at time k the set-point coordinate for the i-th follower is $(r_i^x[k], r_i^y[k])$. Notice that (8) defines $|\mathcal{D}||\mathcal{F}| = 2(n-1)$ optimization problems. More precisely, it defines two independent optimization problems for each follower robot: one for x and one for y. Given that not all followers might have communication with the leader at time k, such optimization problems have to be solved with a distributed algorithm that satisfies the informational constraints imposed by $\mathcal{G}[k]$. In this work, we focus on the design of a distributed algorithm that guarantees that as k goes to infinity, the term $r_i^d[k]$ converges to $c_\ell^d[k] + \delta_i^d[k]$ for all $i \in \mathcal{F}$ and all $d \in \mathcal{D}$, i.e., a convergence in the asymptotic sense. For such matter, we use distributed population dynamics. The second task, on the other hand, is for each robot to actually navigate to its corresponding set-point coordinates. Such set-point coordinates are given by the pre-defined trajectory for the leader robot, and by $(r_i^x[k], r_i^y[k])$ for all $i \in \mathcal{F}$. In this work we focus mainly on the first task, and, in consequence, we use classical PID controllers for the navigation of each robot and we do not consider obstacle avoidance. Nevertheless, under the proposed approach, the local PID controllers can be replaced by any other navigation strategy without having to modify the distributed optimization of the first task. Thus, obstacle avoidance could be added using other navigation controllers such as the one proposed by [34].

To summarize, Fig. 11 presents the control architecture described above. Here, $\mathbf{u}_j[k] \in \mathbb{R}^2$ is a vector that contains the voltages to be applied to the motors of the j-th robot at time k, and $\mathbf{t}_\ell[k] \in \mathcal{X}$ is a vector that contains the reference x and y coordinates for the leader robot at time k. Note that in all cases the navigation task is done using only the local information available to each robot. Regarding the leader, for instance, the local navigation controller uses the reference coordinates given by the pre-defined trajectory, and the current coordinates of the leader robot. Similarly, each follower robot navigates using its own location and the reference coordinates computed distributedly over $\mathcal{G}[k]$. Note that in the case of the leader the distributed controller over $\mathcal{G}[k]$ is not used to provide a reference for the leader's navigation. Instead, it is used to exchange information about the leader's location with the follower robots that can communicate with the leader (the dotted line is used to highlight that in general the leader does not have communication with all followers).

4.3 Distributed Control Over The Network Using Population Dynamics

In this subsection we present our proposed method to solve the distributed optimization problems of (8), which is based on population dynamics [35]. However, we propose a novel discrete-time-class of distributed population dynamics. In the following subsections we present our approach and we develop the corresponding stability analysis of our proposed method.

4.3.1 Distributed Optimization Using Population Dynamics

Consider a set of populations $\mathcal{D}$ that represent the dimensions of $\mathbb{R}^2$, i.e., $\mathcal{D} = \{x,\, y\}$, and let each population be composed by a large and finite number of agents that interact strategically with their population peers. Moreover, let the agents of each population be rational decision makers that, at any time k, select among n different strategies representing the set of robots, i.e., at any time k the set of available strategies is $\mathcal{V} = \{\ell,\, 2,\, \cdots,\, n\}$. To solve the optimization problems of (8) using population dynamics, we interpret the optimization variable $r_i^d[k]$ as the portion of agents of population $d \in \mathcal{D}$ that select the robot $i \in \mathbb{R}$ at time k. Hence, at any time k, the vector $\mathbf{r}^d[k] = \begin{bmatrix} r_\ell^d[k], & r_2^d[k], & \cdots, & r_n^d[k] \end{bmatrix}^\top$ describes the state of population $d \in \mathcal{D}$. Thus, $\mathbf{r}^d[k]$ contains the values of all the optimization variables of (8), as well as the value of an additional variable $r_\ell^d[k]$. Such additional variable does not appear in any of the objective functions of (8) and is not used for the leader's navigation. Instead, the variable $r_\ell^d[k]$ plays the role of an auxiliary variable that allows us to deal with certain invariance properties of the population dynamics (see Remark 1 in Section 4.3.2). Clearly, depending on the mechanism that population agents use to select robots at time k, different trajectories of the population state $\mathbf{r}^d[\cdot]$ emerge on the population $d \in \mathcal{D}$. Therefore, to solve the optimization problems of (8) using this framework, we have to design the strategic interaction of the population agents so that the dynamic evolution of $\mathbf{r}^d[\cdot]$ converges to the minimizer of (8) for all $d \in \mathcal{D}$. The Smith dynamics are one of the fundamental population dynamics and [36] have proposed a continuous-time distributed version of such dynamics. In this work, we build upon such distributed dynamics, but we formulate them with two fundamental differences: (i) we use a discrete-time formulation to obtain stability guarantees for practical implementations where computations are necessarily discrete; and (ii) we saturate the dynamics to obtain less conservative bounds in our theoretical analyses. More precisely, our proposed discrete-time distributed Smith dynamics are given by

$$r_i^d[k+1] = r_i^d[k] + \epsilon^d \sum_{j \in \mathcal{N}_i[k]} \big(f_i^d - f_j^d\big)\theta_{ij}^d \phi_{ij}^d, \tag{9}$$

for all $i \in \mathcal{V}$ and all $d \in \mathcal{D}$, where the scalar $\epsilon^d > 0$ is the step size of the update and is assumed equal for all $i \in \mathcal{V}$; and where

$$\begin{aligned}
f_i^d &= \delta_i^d[k] - r_i^d[k], \quad \forall i \in \mathcal{F} \\
f_\ell^d &= -c_\ell^d[k] \\
\theta_{ij}^d &= \begin{cases} \beta^d, & \text{if } f_i^d = f_j^d \\ \min\big(r_j^d[k],\, \beta^d\big), & \text{if } f_i^d > f_j^d \\ \min\big(r_i^d[k],\, \beta^d\big), & \text{if } f_i^d < f_j^d \end{cases} \\
\phi_{ij}^d &= \begin{cases} 1, & \text{if } |f_i^d - f_j^d| \leq \beta^d \\ \beta^d / |f_i^d - f_j^d|, & \text{if } |f_i^d - f_j^d| > \beta^d. \end{cases}
\end{aligned} \tag{10}$$

Here, the saturations with β^d in θ_{ij}^d and ϕ_{ij}^d are included to obtain less conservative conditions for stability (see Remark 3 in Section 4.3.2). In addition, note that the term ϕ_{ij}^d is always well-defined, and, in fact, $0 \leq \phi_{ij}^d \leq 1$ for all $i, j \in \mathcal{V}$. On the other hand, the scalar function f_i^d represents the payoff provided by the i-th robot to the d-th population. Such scalar function is denoted as a fitness function and, as shown in (10), depends on local information available to the i-th robot (yet, we omit its arguments to simplify the notation). In consequence, notice that (9) defines a set of n equations where the i-th equation depends only on local information available to the i-th robot. Thus, to obtain a distributed computation of (9), it suffices that each robot $i \in \mathcal{V}$ computes its corresponding i-th equation. Furthermore, note that a particular equilibrium of the dynamics of (9) occurs when $f_i^d = f_j^d$ for all $i, j \in \mathcal{V}$ (by equilibrium we mean that $r_i^d[k+1] = r_i^d[k]$, for all $i \in \mathcal{V}$). Such equilibrium is independent of $r_\ell^d[k]$ and occurs when

$$r_i^d[k] = c_\ell^d[k] + \delta_i^d[k], \quad \forall i \in \mathcal{F}. \tag{11}$$

Therefore, if we guarantee that the dynamics in (9) always converge to (11), then we can use such dynamics to solve (8) in a distributed fashion. The study of such convergence is the topic of the next subsection.

4.3.2 Stability Analysis

In this subsection we provide sufficient conditions to guarantee the asymptotic stability of our proposed discrete-time dynamics (9) to the minimizer of (8). However, before doing so, we need to introduce some additional results. First of all, note that (9) can be written as

$$\mathbf{r}^d[k+1] = \mathbf{r}^d[k] + \epsilon^d \mathbf{L}^d[k]\mathbf{f}^d[k], \quad \forall d \in \mathcal{D}, \tag{12}$$

where $\mathbf{f}^d[k] = [f_i^d]$ is an $n \times 1$ vector that collects all the fitness functions of the d-th population; and $\mathbf{L}^d[k] = [l_{ij}^d]$ is an $n \times n$ matrix whose elements depend on $\mathbf{r}^d[k]$ and $\mathcal{G}[k]$, and are given by

$$\begin{aligned} l_{ii}^d &= \sum_{j \in \mathcal{V}\setminus\{i\}} a_{ij}[k]\theta_{ij}^d\phi_{ij}^d, \quad \forall i \in \mathcal{V}, \forall d \in \mathcal{D} \\ l_{ij}^d &= -a_{ij}[k]\theta_{ij}^d\phi_{ij}^d, \quad \forall i,j \in \mathcal{V}, \forall d \in \mathcal{D}. \end{aligned} \tag{13}$$

Here, the arguments of l_{ii}^d and l_{ij}^d have been omitted to ease notation. Notice that the matrix $\mathbf{L}^d[k]$ is the Laplacian of a dynamical graph $\mathcal{G}^d[k]$ that depends on $\mathbf{r}^d[k]$ and the original graph $\mathcal{G}[k]$, and whose adjacency matrix has the elements $a_{ii}^d[k] = 0$ and $a_{ij}^d[k] = -l_{ij}^d$ for all $i,j \in \mathcal{V}$, with $i \neq j$, and for all $d \in \mathcal{D}$. Furthermore, observe that the matrix form given by (12) implies the following assumption.

Assumption 4.3. *If $a_{ij}[k] = 1$, then the population portions $r_i^d[k]$ and $r_j^d[k]$ are updated at time k, for all $d \in \mathcal{D}$.*

These observations lead to the following results.

Lemma 4.1. *Consider the discrete-time distributed Smith dynamics given by (9), and consider the set*

$$\mathcal{M}^d = \left\{ \mathbf{r}^d[k] \in \mathbb{R}^n \,:\, \sum_{i \in \mathcal{V}} r_i^d[k] = m^d \right\}, \tag{14}$$

where the scalar $m^d > 0$ is the total population mass. If Assumption 3 holds, then the set $\mathcal{M}^d$ is forward invariant under the dynamics (9). That is, $\mathbf{r}^d[k] \in \mathcal{M}^d$ implies that $\mathbf{r}^d[k+1] \in \mathcal{M}^d$ for all k.

Proof. For all $d \in \mathcal{D}$ and all k it holds that

$$\begin{aligned} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i[k]} (f_i^d - f_j^d)\theta_{ij}^d\phi_{ij}^d &= \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} a_{ij}[k](f_i^d - f_j^d)\theta_{ij}^d\phi_{ij}^d \\ &= 0. \end{aligned}$$

This follows from the facts that $a_{ij}[k] = a_{ji}[k]$, $\theta_{ij}^d = \theta_{ji}^d$, and $\phi_{ij}^d = \phi_{ji}^d$, for all $i,j \in \mathcal{V}$, all $d \in \mathcal{D}$, and all k. Therefore, we have that $\sum_{i \in \mathcal{V}} r_i^d[k+1] = \sum_{i \in \mathcal{V}} r_i^d[k]$ for all k, i.e., $\sum_{i \in \mathcal{V}} r_i^d[k] = \sum_{i \in \mathcal{V}} r_i^d[0] = m^d$ for all k. □

Lemma 4.2. *Consider the discrete-time distributed Smith dynamics given by (12), and consider the set*

$$\Delta^d = \mathbb{R}_+^n \cap \mathcal{M}^d, \tag{15}$$

where $\mathbb{R}_+^n$ is the strictly positive orthant of $\mathbb{R}^n$. If Assumption 3 holds and $\mathbf{r}^d[k] \in \Delta^d$ for all k, then the following properties hold:

(i) The matrix $\mathbf{L}^d[k]$ is positive semi-definite for all k.

(ii) The dynamical graph $\mathcal{G}^d[k]$ has the same communication links as the original graph $\mathcal{G}[k]$ for all k.

Proof. To prove (i) we observe that $\mathbf{L}^d[k]$ is real, symmetric, and diagonally dominant (see (13)). In addition, if $\mathbf{r}^d[k] \in \Delta^d$ for all k, then the diagonal elements of $\mathbf{L}^d[k]$ are non-negative for all k. In consequence, $\mathbf{L}^d[k] \succeq 0$ for all k. To prove (ii) we notice that the fitness functions defined in (10) are finite over all Δ^d. In consequence, $\phi_{ij}^d > 0$ for any $\mathbf{r}^d[k] \in \Delta^d$ and for all $i,j \in \mathcal{V}$. Similarly, if $\mathbf{r}^d[k] \in \Delta^d$, we have that $\theta_{ij}^d > 0$ for all $i,j \in \mathcal{V}$. In consequence, $a_{ij}^d[k] > 0$ if and only if $a_{ij}[k] = 1$. Thus, $\mathcal{G}[k]$ and $\mathcal{G}^d[k]$ have the same set of communication links for all k. □

Remark 1: Lemma 4.2 requires the forward invariance of the set Δ^d given in (15). Note that such invariance implies that $r_i^d[k] \in \mathbb{R}_+$ and $\sum_{i\in\mathcal{F}} r_i^d[k] = m^d - r_\ell^d[k]$, for all $i \in \mathcal{V}$, all $d \in \mathcal{D}$, and all k. Hence, $\sum_{i\in\mathcal{F}} r_i^d[k] < m^d$, for all $d \in \mathcal{D}$ and all k. Thus, to guarantee that the reference coordinate $(r_i^x[k], r_i^y[k])$, for all $i \in \mathcal{F}$, can achieve any value within $\mathcal{X}$, the population masses m^x and m^y have to be big enough. Given that $m^d = \sum_{i\in\mathcal{V}} r_i^d[0]$ for all $d \in \mathcal{D}$, in order to guarantee that any formation within $\mathcal{X}$ is attainable, it suffices to satisfy the following condition:

$$\begin{aligned} &\sum_{i\in\mathcal{V}} r_i^d[0] \geq n\beta^d, \quad \forall d \in \mathcal{D}, \\ &r_i^d[0] > 0, \quad \forall i \in \mathcal{V}, \forall d \in \mathcal{D}, \end{aligned} \tag{16}$$

where β^d is the spatial length of the d-th dimension of the rectangle $\mathcal{X}$ defined in (7). Note that if the initial position of the robots is within $\mathcal{X}$, then condition (16) can be satisfied under the informational constraints of $\mathcal{G}[k]$ by setting $r_\ell^d[0] \geq n\beta^d$ for all $d \in \mathcal{D}$, and $r_i^x[0] = x_i[0]$ and $r_i^y[0] = y_i[0]$ for all $i \in \mathcal{F}$.

With the aid of these preliminary results, we now provide sufficient conditions that guarantee the asymptotic stability of the minimizers of (8) under the proposed method. The next theorem illustrates our result.

Theorem 4.1. *Consider the discrete-time distributed Smith dynamics given in (9) and (12). Let $\tilde{n}$ denote the maximum number of robots that any robot can communicate with (simultaneously) at any time k. Moreover, supose that Assumptions 1, 2, and 3 hold, and let the following conditions be satisfied:*

(i) The initial population state satisfies condition (16), i.e., $\mathbf{r}^d[0] \in \mathbb{R}_+^n$ and $\sum_{i\in\mathcal{V}} r_i^d[0] \geq n\beta^d$, for all $d \in \mathcal{D}$.

(ii) For all $d \in \mathcal{D}$, it holds that $0 < \epsilon^d < 1/(\tilde{n}\beta^d)$.

Then the minimizers of (8) are asymptotically stable under the proposed discrete-time distributed Smith dynamics.

Proof. First, to induce the results of Lemma 4.2, we prove that under the given conditions it holds that $\mathbf{r}^d[k] \in \Delta^d$ for all k, where Δ^d is given by (15) with $m^d \geq n\beta^d$. From Lemma 4.1 we can conclude that $\mathbf{r}^d[k] \in \mathcal{M}^d$ for all k. Thus, we have to prove only the positiveness of $\mathbf{r}^d[k]$ for all k. To do so, given that by assumption $\mathbf{r}^d[0] \in \mathbb{R}_+^n$, we have to prevent any sign changes in $r_i^d[k]$ for all $i \in \mathcal{V}$ and all k. From (9)-(10) we note that the only terms of the summation in (9) that favor the decrement of $r_i^d[k]$ are those where $f_i^d - f_j^d < 0$. Thus, we should consider only the critical (impossible) case where $f_i^d < f_j^d$ for all $i, j \in \mathcal{V}$. Under such scenario we have that (9) can be written as

$$r_i^d[k+1] = r_i^d[k] - \epsilon^d \sum_{j\in\mathcal{N}_i[k]} \left|f_i^d - f_j^d\right| \min(r_i^d[k], \beta^d)\phi_{ij}^d,$$

for all $i \in \mathcal{V}$ and all $d \in \mathcal{D}$. Moreover, note that $r_i^d[k] \geq \min(r_i^d[k], \beta^d)$ for all k. Hence, to obtain a sufficient condition that prevents sign changes and guarantees positiveness, without loss of generality we can consider the case where θ_{ij}^d does not have a $\min(\cdot)$ saturator, i.e., the case where $\theta_{ij}^d = r_i^d[k]$ for any $\mathbf{r}^d[k] \in \Delta^d$ where $f_i^d < f_j^d$. Furthermore, we can also focus only on the worst (impossible) case where $|f_i - f_j| = \beta^d$ for all $i, j \in \mathcal{V}$ (such saturation is given by the term ϕ_{ij}^d). Joining all these critical cases we get that (9) can be written as

$$\begin{aligned} r_i^d[k+1] &= r_i^d[k]\Big(1 - \epsilon^d \sum_{j\in\mathcal{N}_i[k]} \beta^d\Big), \quad \forall i \in \mathcal{V}, \forall d \in \mathcal{D} \\ &= r_i^d[k]\big(1 - \epsilon^d \tilde{n}\beta^d\big), \quad \forall i \in \mathcal{V}, \forall d \in \mathcal{D}. \end{aligned}$$

Therefore, to prevent sign changes of $r_i^d[k]$ we require that $0 < \epsilon^d < 1/(\tilde{n}\beta^d)$, which is guaranteed by condition (ii). Observe that the strict bounds in this condition not only prevent sign changes, but also prevent that $r_i^d[k]$ becomes zero for any k. Thus, by conditions (i) and (ii) it holds that $\mathbf{r}^d[k] \in \Delta^d$ for all k, and Lemma 4.2 applies. From now on, we consider the original dynamics given by (9) and not only the worst cases, i.e., we consider the saturator in θ_{ij}^d.

Now that we have proved that Lemma 4.2 applies, we can proceed to prove the asymptotic stability of the proposed dynamics. For that, notice that the dynamics of (9) are invariant under the addition of a scalar to the fitness functions. This follows from the fact that $(f_i^d - f_j^d) = (f_i^d + \alpha - f_j^d - \alpha)$ for any $\alpha \in \mathbb{R}$. Thus, the dynamics under the original fitness vector $\mathbf{f}^d[k]$ are exactly the same as if the fitness vector $\tilde{\mathbf{f}}^d[k] = \mathbf{f}^d[k] + c_\ell^d[k]\mathbf{1}_n$ were used instead (here $\mathbf{1}_n$ denotes a column vector with n ones), i.e., we have that $\tilde{f}_\ell^d = 0$ and $\tilde{f}_i^d = c_\ell^d[k] + \delta_i^d[k] - r_i^d[k]$ for all $i \in \mathcal{F}$. In consequence, without loss of generality, we analyze the stability of $\mathbf{r}^d[k+1] = \mathbf{r}^d[k] + \epsilon^d \mathbf{L}^d[k]\tilde{\mathbf{f}}^d[k]$ to obtain convergence results that are also valid for (12). The advantage of using $\tilde{\mathbf{f}}^d[k]$, instead of $\mathbf{f}^d[k]$, is that $\tilde{\mathbf{f}}^d[k]$ can be written as $\tilde{\mathbf{f}}^d[k] = -\mathbf{B}\mathbf{e}^d[k]$, where $\mathbf{B}$ is an $n \times n$ diagonal matrix with the (ℓ,ℓ)-th element equal to zero and the other diagonal elements equal to 1; and $\mathbf{e}^d[k] = \mathbf{r}^d[k] - \mathbf{r}_*^d[k]$, where $\mathbf{r}_*^d[k]$ is the population state for which $\tilde{f}_i^d = \tilde{f}_j^d = 0$ for all $i, j \in \mathcal{V}$, i.e., the minimizer of (8). Moreover, from Assumption 2 we can conclude that such population state $\mathbf{r}_*^d[k]$ is always attainable with a population mass $m^d \geq n\beta^d$ (see condition (i)), i.e., $\mathbf{r}_*^d[k]$ is always within Δ^d. Under this formulation, we can use the function $V(\mathbf{r}^d[k]) = (\mathbf{e}^d[k])^\top \mathbf{B}\mathbf{e}^d[k]$ as a valid Lyapunov function candidate (i.e., $V(\mathbf{r}^d[k]) > 0$, for all $\mathbf{r}^d[k] \neq \mathbf{r}_*^d[k]$ and $V(\mathbf{r}^d[k]) = 0 \iff \mathbf{r}^d[k] = \mathbf{r}_*^d[k]$). It is worth mentioning that although $\mathbf{B}$ is a positive semi-definite matrix, the proposed Lyapunov function is positive definite over Δ^d. To see why, note that $\mathbf{B}$ has only one zero diagonal element and such element is associated to $r_\ell^d[k]$. Nevertheless, by Lemma 4.1 we have that $r_\ell^d[k] = m^d - \sum_{i\in\mathcal{F}} r_i^d[k]$ for all k. Thus, there are only $n-1$ independent population portions, and, in consequence, the line where $V(\mathbf{r}^d[k]) = 0$ contracts to a single point in $\mathbb{R}^n$ for every k. Continuing with the proof, for $\mathbf{r}_*^d[k]$ to be asymptotically stable we require that $V(\mathbf{r}^d[k+1]) - V(\mathbf{r}^d[k]) < 0$ for all k. If we denote $\tilde{V} = V(\mathbf{r}^d[k+1]) - V(\mathbf{r}^d[k])$ we get:

$$\tilde{V} = \left(\mathbf{e}^d + \epsilon^d \mathbf{L}^d \tilde{\mathbf{f}}^d\right)^\top \mathbf{B}\left(\mathbf{e}^d + \epsilon^d \mathbf{L}^d \tilde{\mathbf{f}}^d\right) - (\mathbf{e}^d)^\top \mathbf{B}\mathbf{e}^d,$$

where we have removed the time index in $\mathbf{e}^d[k]$, $\mathbf{L}^d[k]$, and $\tilde{\mathbf{f}}^d[k]$ as all the terms are now at time k. Solving this expression we get that $\tilde{V}$ equals:

$$\begin{aligned}
&= \epsilon^d (\mathbf{e}^d)^\top \mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d + \epsilon^d (\mathbf{L}^d\tilde{\mathbf{f}}^d)^\top \mathbf{B}\mathbf{e}^d + (\epsilon^d)^2 (\mathbf{L}^d\tilde{\mathbf{f}}^d)^\top \mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d \\
&= \epsilon^d (\mathbf{B}\mathbf{e}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d + \epsilon^d (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d \mathbf{B}\mathbf{e}^d + (\epsilon^d)^2 (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d \\
&= -\epsilon^d (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d - \epsilon^d (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d + (\epsilon^d)^2 (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d \\
&= -2\epsilon^d (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d + (\epsilon^d)^2 (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d,
\end{aligned}$$

where we have used the facts that $\mathbf{B} = (\mathbf{B})^\top$, $\mathbf{L}^d = (\mathbf{L}^d)^\top$, and $\tilde{\mathbf{f}}^d = -\mathbf{B}\mathbf{e}^d$. Notice that due to the form of $\mathbf{B}$ it holds that $(\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d \leq (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\mathbf{L}^d\tilde{\mathbf{f}}^d$. Thus, setting $\mathbf{L}^d = \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}$ (by spectral decomposition) we have that

$$\begin{aligned}
(\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\mathbf{B}\mathbf{L}^d\tilde{\mathbf{f}}^d &\leq (\tilde{\mathbf{f}}^d)^\top \mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}\mathbf{P}\mathbf{\Lambda}\mathbf{P}^{-1}\tilde{\mathbf{f}}^d \\
&\leq (\tilde{\mathbf{f}}^d)^\top \mathbf{P}\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}\mathbf{\Lambda}^{1/2}\mathbf{P}^{-1}\tilde{\mathbf{f}}^d \\
&\leq \lambda_{max}^{\mathbf{\Lambda}} (\tilde{\mathbf{f}}^d)^\top \mathbf{P}\mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}\mathbf{P}^{-1}\tilde{\mathbf{f}}^d \\
&\leq \lambda_{max}^{\mathbf{L}^d} (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d.
\end{aligned}$$

Here $\lambda_{max}^{\mathbf{\Lambda}} = \lambda_{max}^{\mathbf{L}^d}$ is the maximum eigenvalue of $\mathbf{L}^d$, and we have used the fact that a quadratic form $\mathbf{x}^\top\mathbf{Z}\mathbf{x}$ with symmetric $\mathbf{Z}$ is always upper-bounded by $\lambda_{max}^{\mathbf{Z}}\mathbf{x}^\top\mathbf{x}$. From the assumption that $\mathcal{G}[k]$ is connected, and due to Lemma 4.2, we can conclude that $\mathcal{G}^d[k]$ is also connected. This means that $\mathbf{L}^d$ has only one eigenvalue equal to zero, and, from Lemma 4.2, we have that the remaining eigenvalues are strictly positive. Thus, $\lambda_{max}^{\mathbf{L}^d} > 0$. Moreover, from Gershgorin Circle Theorem we have that $\lambda_{max}^{\mathbf{L}^d} \leq 2\max_{i\in\mathcal{V}} l_{ii}^d$, and from (13) we can conclude that $\lambda_{max}^{\mathbf{L}^d} \leq 2\tilde{n}\beta^d$. Hence,

$$\begin{aligned}
\tilde{V} &\leq -2\epsilon^d (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d + 2\tilde{n}\beta^d(\epsilon^d)^2 (\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d \\
&\leq 2\epsilon^d(\tilde{n}\beta^d\epsilon^d - 1)(\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d.
\end{aligned}$$

Notice that $(\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d$ is the quadratic form of the Laplacian of $\mathcal{G}^d[k]$. Given that $\mathcal{G}^d[k]$ is undirected and connected (by the symmetry of $\mathbf{L}^d$ and Lemma 4.2), we can conclude that $(\tilde{\mathbf{f}}^d)^\top \mathbf{L}^d\tilde{\mathbf{f}}^d$ is always positive and

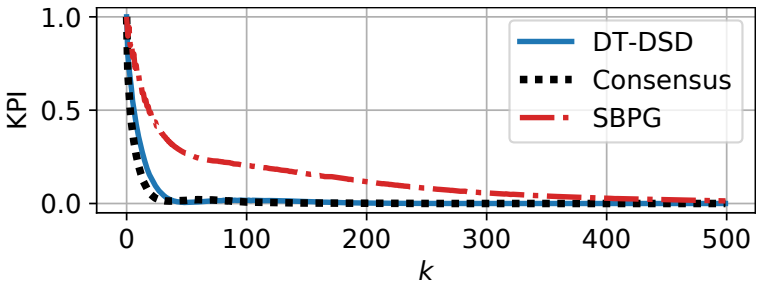


Figure 12: Experimental results on real robots.

is zero only when $\tilde{\mathbf{f}}^d \in \text{span}(\mathbf{1}_n)$, i.e., when $f_i^d = f_j^d$ for all $i, j \in \mathcal{V}$. Furthermore, if ϵ^d satisfies condition (ii), the term $(\tilde{n}\beta^d\epsilon^d - 1)$ is always strictly negative. In consequence, $\tilde{V}$ is always non-positive and is zero only when $\mathbf{r}^d[k] = \mathbf{r}_*^d[k]$. Thus, $\mathbf{r}_*^d[k]$ is asymptotically stable. □

Remark 2: Observe that Theorem 4.1 is valid for any connected and undirected graph topology. Moreover, in the proof of Theorem 4.1 the Laplacian of $\mathcal{G}^d[k]$ only appears at time k. By induction, Theorem 4.1 applies unchanged for time-varying graphs that remain connected and undirected for all k.

Remark 3: Note that without the saturations in θ_{ij}^d and ϕ_{ij}^d, the term $\lambda_{max}^{\mathbf{L}^d} \leq 2\max_{i\in\mathcal{V}} l_{ii}^d$ would be upper-bounded by $2\tilde{n}n\beta^d$ and stability would not be guaranteed by condition (ii) (it can be shown that a similar issue occurs with the bound required for positiveness). However, given that the formations are assumed to lie within $\mathcal{X}$, the saturations in (10) allow us to consider only the region of interest and obtain less conservative bounds for ϵ^d.

Remark 4: Notice that the conjunction of Assumptions 1 and 3 implies that all robots update their corresponding variables at all times k. Thus, Theorem 4.1 assumes that there is a persistent synchronicity on the robots' operation, which might be hard to satisfy for some robotic applications. Although we leave the formal study of asynchronous population dynamics for a future work, in Section 4.3.2 we provide some experiments where the graph $\mathcal{G}[k]$ is not connected for all k, and so the synchronicity of all robots is not required for all k (the synchronicity is required only for each connected sub-graph at each time k). Such experiments illustrate the fact that the aforementioned persistent synchronicity of all robots is only a sufficient condition for asymptotic stability.

In this subsection we have illustrated the theoretical developements of this second part on a real multi-robot platform comprised by a set of six e-puck v2 robots. We have only show the results of a particular experiment under a time-invariant path graph (i.e., $\tilde{n} = 2$), which is the connected graph with the most distributed topology. However, videos of several other experiments are available at `youtu.be/t-S0Gtblh_A` and some of them consider time-varying graphs that are not connected for all k. The results of our particular experiment are depicted in Fig. 12 where DT-DSD denotes our discrete-time distributed Smith dynamics; Consensus refers to the consensus algorithm of [27]; and SBPG denotes the state-based potential games approach proposed by [31]. As key performance index (KPI) we have taken the sum of the objectives functions of (8) over all $i \in \mathcal{F}$ and all $d \in \mathcal{D}$, and we have normalized it up to 1. In all cases, robots start in the same initial position and the required formation is an hexagon. The bounds for the DT-DSD are taken according to Theorem 4.1 with $\beta^x = 125$ and $\beta^y = 90$ (the dimensions of our testbed); the bound for the Consensus is taken according to [27]; and the bound of SBPG is set at the biggest value (within 0.001 precision) that converged in this particular experiment. As shown in Fig. 12, all the methods converge to the desired solution. In particular, to achieve a KPI lower than 0.02 our method required 37 iterations, Consensus required 28, and SBPG required 451 (iterations were taken every 0.1 seconds). Thus, we can conclude that our method outperforms the SBPG and obtains a performance comparable to the consensus algorithm. The advantage of our method in contrast with Consensus, is that it preserves the forward-invariance of Δ^d, and, in consequence, is suitable not only to achieve robotic formations as in this section, but also for resource allocation problems that are relevant for many other distributed optimization and control applications (see [32]).

5 Distributed multiagent software development for attacks and dynamic testing

This section describes the software we developed to implement a decentralized control. This software allows users to configure several parameters to generate results for different formations.

5.1 Software description

The designed code works on a way that any person, no matter its knowledge about the described algorithms, may configure and run different simulations with the net of robots.

Initially, as its shown in Figure 13, the program has five principal options. The first option gives users a brief description of the system and the principal knowledge that they need to implement the system on the platform with the robots as well as some general recommendation to implement the proposed idea on other contexts. The second option allows users to calibrate the camera filters in order to distinguish the robots and their positions during the simulation. The third option allows users to configure communications among the agents using the IP protocol. The fourth option is rather practical as it permits running different type of dynamics with previous configurations. Finally, the fifth option runs the simulation of the system.

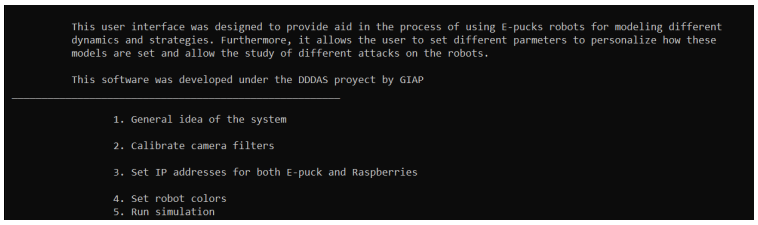


Figure 13: Initial interface of the program.

The general architecture of the system forces the user to calibrate the camera, and the robots or raspberries IP at least once. This allows the system to initialize all required elements even if the console running the code stops working somehow by means of pickle files. Once the simulation starts to run, it is necessary to describe the two more important parameters: the adjacency matrix and the desired formation, as it is shown in Figure 14, where the software interface describes the importance of these parameters.

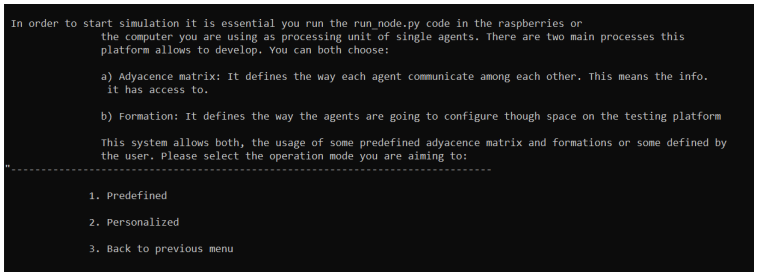


Figure 14: Initial parameters need on the code.

The user may utilize predefined models, like the ones discussed in the previous sections, or set a new model. When an adjacency matrix has to be defined, the system gives eight possible options indistinctly of the number of agents involved, so the user can select the one that he needs the most. Additionally, the system has ten different predefined formations for the case with six agents, so that the user can easily select one of these formations. Also, the user can define a new formation by setting δx and δy distances from agents to the leader.

Currently, the platform supports direct initialization for the following adjacency matrices and formations:

Formations	Adjacency Matrices
G formation	Complete
I formation	Empty
A formation	Regular Ring Lattice
P formation	Cycle
GIAP formation	Path
Diagonal formation	Erdos Reyni
Triangle formation	Watts Strogatz
Horizontal line formation	Barabasi
Vertical formation	
Hexagonal formation	

Table 5: Supported predefined formations and adjacency matrices

5.2 Results

The system code is available at Github: it contains the interface code[3], which was developed to make it easier for any user to configure different types of communication approaches and formations. In general, simulations developed in previous sections were replicated using different numbers of agents to test the system actual capacity[4].

Communication protocols among Raspberry-pi and E-puck robots performed effectively. However, the dependence of the TCP/IP protocol leads to a yet unsolved problem; the need of being aware of different IPs in order to be able to set the connection up, which in turn might be a problem for different contexts in which many devices are connected to same Access Point. This was partly addressed by considering the central computer as a server and the rest of devices as clients. Therefore, besides the efforts for developing a plug and play system, some configurations are yet to be made through the console of the systems, in particular the selection of the correct IP address from the central station.

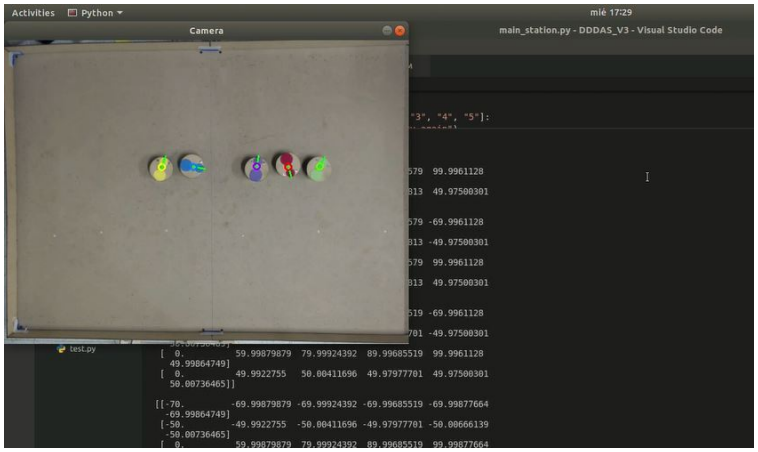


Figure 15: Sample of formation achieved using interface.

Furthermore, consensus algorithms developed by association matrix approach and calculated directly on each raspberry as well as further connection with e-pucks was achieved. Even though the main objective was to obtain a decentralized system, there is still the need to use some centralized information from the camera, as well as to monitor and visualize the whole system. This means that we still need to calibrate the system every time some information of one of its agents changes (e.g., color or the IP address). However, the interface allows to perform this process easily[5]

[3]Available at `https://github.com/idperez720/DDDAS_V3`

[4]For instance, 5 agents hexagonal formation implementation can be seen at `https://youtu.be/aETW-ARM6K4`

[5]This can be seen in the following vide. Sample of system calibration and predefined dynamics implementation: `https:`

Finally, regarding activities iii), v), ix), and x) were completed successfully. Parameter selections, as previously stated, allowed the user to select any formation that can be defined by means of the distances between any number of agents. This leads to another limitation which is the impossibility of changing dynamically the association matrix for implementing time-varying communications networks. Mechanisms for the detection of anomalies were implemented as well.

5.3 General recommendations for interface usage

Despite the fact that the original purpose of this software is to provide an interface to ease the implementation of different formations, dynamics, and algorithms, it does have some relevant limitations that must be taken into consideration for achieving reliable results.

The first relevant limitation is that this platform does not provide any support for other robots rather than E-pucks reported in previous sections. The main issue remains on how communication protocol works with these models, which means available code would be unable to set, configure, and control other models available in the market. This problem remains unsolved by the time this version was uploaded to provided git repository. However, further implementation of this approach on other models is feasible but its out of the scope of this development.

On the other hand, as the camera connects the main station through a server (that allows the main station to calculate the position of the robots that are set in the platform used), the system is quality-dependent on camera resolution. Furthermore, platform where robots interact must have:

(i) Defined area limits.

(ii) Constant illumination throughout its surface.

(iii) Different background color from the ones used to identify the robots.

These limitations are set mainly due to the image processing algorithm as HSV color space is used. In general it is further recommended to notice different colors next to the platforms from other objects such as tapes placed on the floor as camera usually records not only the platform itself but some region beyond its borders. Any color similar to the ones being used inside the platform must be removed as it can directly affect system performance. Fortunately, calibration algorithms developed allow to test if there is any problem with robot recognition process, which means this potential issue can be debugged by developing a careful calibration process.

5.4 Conclusion of the implementation

To summarize, it was possible to complete the requested work achieving the expected objectives. Firstly, it was possible to develop and implement in software, the control algorithm for the mitigation of attacks under different conditions of exchange information between the agents. In addition, it was possible to take these tests to a robots platform where it is possible visualize physically the behavior of the agents, thus approaching a real implementation of the control techniques investigated and developed. Finally, the development and implementation of this platform allows the possibility for future students and researchers to continue working with it.

6 Extended DDDAS System

6.1 Introduction

Use of autonomous vehicles (AVs) has increased since their introduction in 2010. This technology is been explored by several companies; Google has been working on self-driving cars with the goal of having easier and safer driving systems [37] and Amazon has developed autonomous vehicles for delivery [38]. AVs are also used for monitoring varios rural and residential areas [39]. As a consequence of these developments, the

//youtu.be/T_JwglIf5vY

United States Department of Transportation is considering advantages and limitations of automated safety technologies [40].

AVs collect and transmit data that later are used to make decisions; decisions based on data may be done online, by automated systems, or offline, by humans and by statistically-based systems. However, the growth in the use of this technology has increased its issues regarding information control, including authentication, integrity and safety [39].

6.2 Problem Description

The systems based on AVs must address different data security requirements; depending on the context, these requirements include confidentiality, integrity, availability or a combination of them.

To satisfy security requirements, AV systems implement several mechanisms like cryptography. Cryptography supports integrity and confidentiality of transmitted and stored data. However, there are additional security issues: first, sensors of the AVs may be compromised and AVs may use automated support while they perform their functions. To address the first issue, we proposed a system that may detect and correct non-expected values. This section addresses the second issue; it proposes a system to improve integrity of data collected to be used to make automated decisions and support other analyses.

6.3 Proposal

We propose a a system, based on SDN and blockchain, to guarantee integrity of data collected by AVs and stored to make automated decisions. First, we present the main features and advantages of the related technology: SDN and blockchain.

6.3.1 SDN.

SDN enables network reconfiguration using a programmatic approach. The advantages of SDN have already been recognized in traditional information technology networks [41] and are also being explored in other kind of systems like Cyberphysical systems [42, 43], wireless networks [44], cellular networks [45], wireless sensor networks and wireless mesh networks [44].

SDN advantages originate from two key points of the paradigm: i) The possibility of using commercial Off-the Shelf generic servers, instead of proprietary hardware and middleboxes with limited software programmability, to support network operations and ii) SDN decouples network control plane from network data plane; the former is responsible of making decisions and the latter of forwarding data accordingly. This separation enables automatic network reconfiguration as an answer to changes in traffic. Additionally, SDN has an application plane, a set of applications that receive, analyze and synthesize data that may be later used as system feedback. Figure 16 presents an overview of a SDN architecture; it includes three main layers: applications, controller and infrastructure.

SDN enables dynamic and automatic network management by extending network architectures with agents, that collect data about actual network state, and a central component (an SDN controller), that receives data from the agents and adjusts network behavior accordingly. SDN makes it easier to develop new network services. While traditional networks process traffic through dedicated hardware devices that are difficult to reprogram and unable to scale services to variable demand, SDN can easily change network behavior and scale services dynamically.

In our previous work [46, 47], we leveraged SDN to enhance Industrial Control Systems (ICS) Security. It uses an SDN controller that evaluates values reported by sensors against values generated in a controlled environment, like the simulation previously mentioned. The controller uses such evaluation to decide whether the received values are within an expected range or not and reacts to that by changing network configuration, isolating sensors, redirecting traffic to honeypots, etc.

In this work, we propose using SDN to enhance our DDDAS system. SDN can build a global view of the network configuration and traffic load and can dynamically change such configuration according to collected data, actual traffic patterns, and even predefined quality of service (QoS) policies to give priority to specific endpoints. Our goal is to integrate features of DDDAS and SDN to build a system that detects incidents and dynamically and automatically reacts to them.

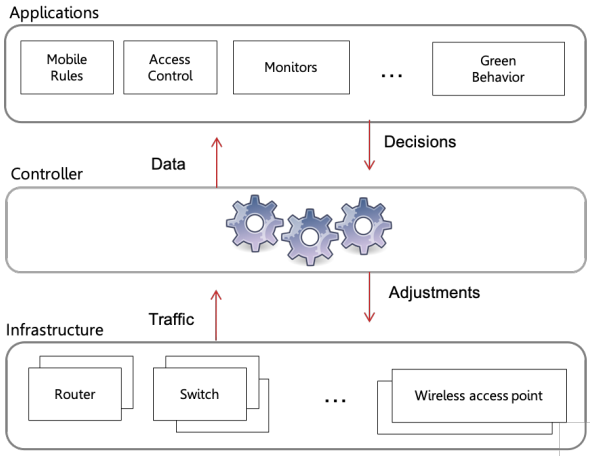


Figure 16: SDN Architecture. SDN has three layers: Applications, Controller and Infrastructure, each one with different functions.

6.3.2 Blockchain.

Blockchain is a distributed database that records information and offers data tracing and immutability. In this kind of system all participants may have a local copy of stored data, thus all records are public, although identification of data issuers may be confidential. Blockchain uses a peer-to-peer approach where every participant follows the same protocol to add and validate transactions. Figure 16 presents an overview of a blockchain architecture; in this example all participants have a local copy of the data.

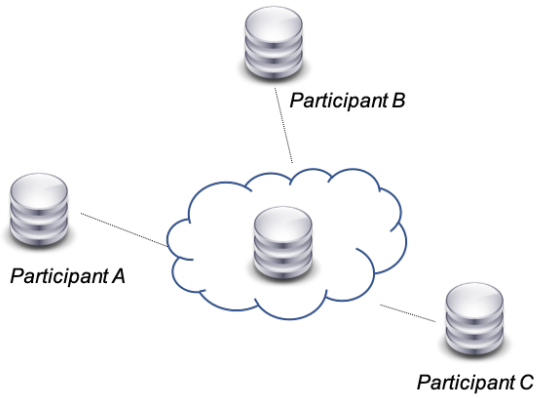


Figure 17: Blockchain Configuration. Multiple participants have local copies of data.

In blockchain, data additions are organized as transactions, several transactions are packed in a block and blocks are linked giving them chronological order. It guarantees immutability because the protocol provides a way of guaranteeing that once a transaction is added to the system, updates are computationally very expensive thus non-viable.

Blockchain adds new transactions to the most recent block and when a given number of transactions is reached it starts a consensus protocol; the consensus validates the transactions in the block, adds the cryptographic hash code of the previous block and generates a new cryptographic hash code for the last block. The generation of the cryptographic hash code is built in such a way that it would be very expensive to try to change stored data. There are several schemes to generate the cryptographic hash code; Proof of Work (PoW) is one of them, it requires a considerable number of computations while looking for a code that meets certain properties. When that code is found, it is added to the block, the block is considered secured, and a new block is created. The linked cryptographic hash codes make it possible to validate the

chain of blocks, and the transactions within each block. The task of looking for a hash code with particular properties is know as mining. Because of the high cost of mining, some blockchain developers proposed having a different consensus protocol, thus creating private blockchains. Public blockchains use PoW as the consensus protocol, while private blockchains use other less expensive consensus protocols.

A private blockchain requires a central authority to validate participants and transactions. The central authority is responsible of validating every transaction, reduces response times and allows handling permissions to support privacy, confidentiality and integrity requirements. Although including a central authority, as private blockchains require, changes the original blockchain scheme to support immutability, private blockchains have the previously mentioned advantages and different businesses are exploring their actual scope.

6.4 Design

This section describes the architecture of the extended DDDAS system; the extended architecture adds an application at the SDN application layer that uses blockchain to guarantee data integrity. The goal is to record data generated by the controller and the autonomous vehicles (AVs) to be able to evaluate behavior across several deployments. The issue that is being addressed is the lack of trusted logs to identify sensor errors or compromises and correct the problems of deficient sensors between deployments of AVs. This extension does not replace the functionality of the main station, its function is complementary; the main station continues monitoring online behavior and generating online adjustments during AVs deployments.

Figure 18 shows the system with the extension in blue:

- The main station plays the role of a SDN controller, receiving data and making online adjustments. It must register adjustments in the blockchain.
- AVs behave as infrastructure devices. Also, they must register their measurements in the blockchain.
- The Data Analysis Application uses previously registered data to detect errors across several deployments and generates error alerts to an operator.

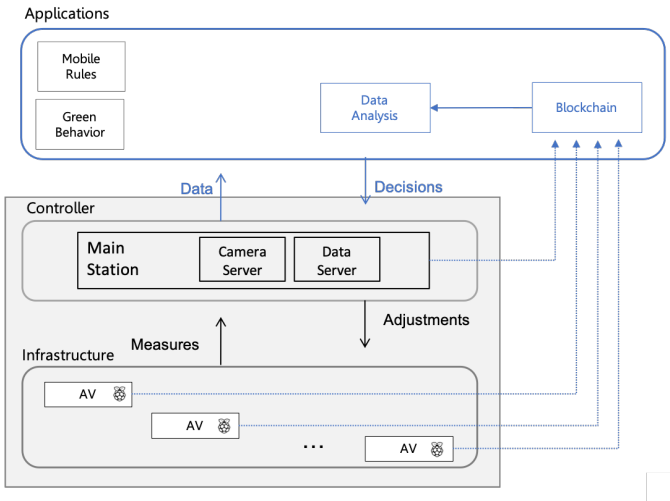


Figure 18: Extended DDDAS Architecture. The extension includes the main station, that acts as controller, and an application, based on blockchain, to collect and analyze data across several deployments of the system.

Although, in our design we wanted AVs to register data in the blockchain, in practice AVs must not report data directly to the blockchain. AVs have constrained resources and blockchain is not time efficient to complete requests. As a consequence, each AV must delegate data registration to an agent that acts on its behalf; an agent receives the data from the AVs and interacts with the blockchain to finish the request.

The addition of agents that act of behalf of the AVs may var; we may have one agent per AV or one agent to represent several AVs. Figure 19 illustrates the former, while Figure 20 illustrates the later.

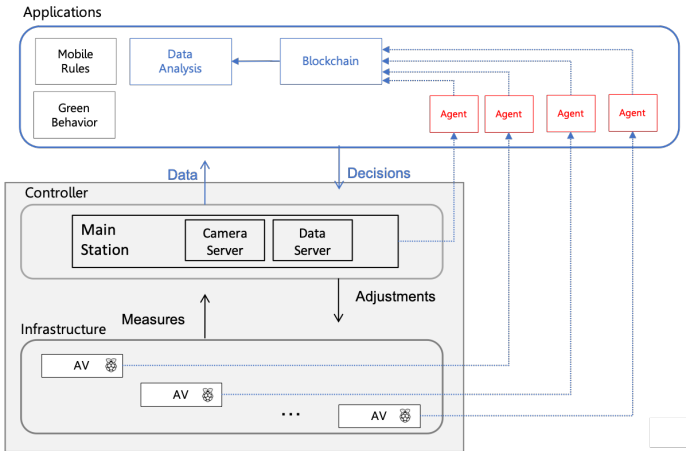


Figure 19: Extended DDDAS Architecture with an agent acting as a proxy for each AV.

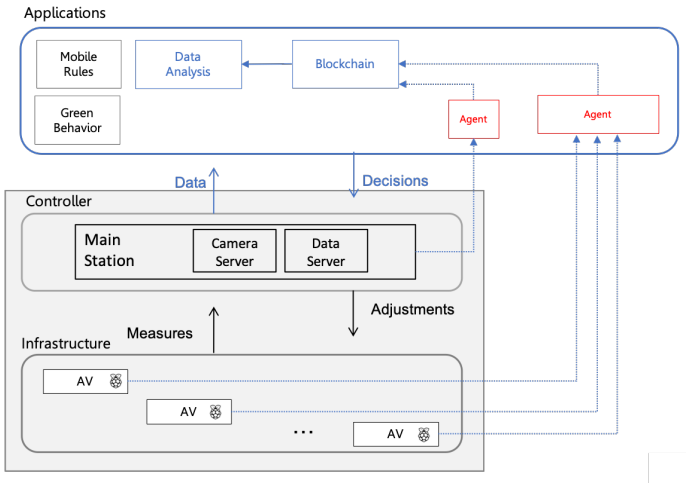


Figure 20: Extended DDDAS Architecture with a single agent acting as proxy for all the AVs

6.5 Implementation

For our implementation we selected Hyperledger[6], a private blockchain, because private blockchains have a better performance time than public blockchains. Also, private blockchains allow an administrator to control who joins the peer-to-peer network and who has access to data. These characteristics are adequate for our DDDAS system.

Hyperledger configuration requires the following steps:

- *Network Configuration.* Network structure depends on context. In this case, we need a Hyperledger network with:

  (i) A central authority (CA). Hyperledger has a Membership Service Provider (MSP) that handles identities, certificates, user authentication and associated cryptographic mechanisms. This authority defines the valid members of a blockchain. In practice, several MSPs may join the same blockchain; each MSP represents and handles membership for a given organization. Our DDDAS environment needs one single MSP.

  (ii) One peer per agent (one agent may represent one or several AVs). A peer keeps a local copy of the data and executes programs to register new transactions on the blockchain.

[6]https://www.hyperledger.org/

(iii) An ordering service that determines the order in which transactions must be registered in the blockchain; it is recommended to have 3 to 5 ordering nodes to support an ordering service.

(iv) One channel. Hyperledger allows having several different network overlays over the same blockchain network; all the members of a given overlay must share the same channel. Our DDDAS environment does not need different overlays thus, we only have one channel.

- *Resource Provisioning.* Resource requirements depend on the number and frequency of transactions. A peer requires enough RAM to process transactions and enouch disk space to store a local copy of the blockchain.
- *Central authority (CA) configuration.* A blockchain network requires one MSP and one administrator. A MSP needs a self-signed certificate (X.509) that becomes the root of trust; the foundation used to certificate the identity of all entities that belong to the same blockchain. It is also used to generate the certificate of the MSP administrator and the list of certificates of the valid members.
- *Peer Deployment.* After the previous configuration steps, the blockchain network is ready to run.

The resulting Hyperledger network has one MSP service, one ordering service, one channel and one peer associated to the main station. In addition, the network may have one or more peers representing the AVs. Each AV corresponds to one epuck and its Raspberry Pi. Figure 21 illustrates a configuration where each epuck has its own agent.

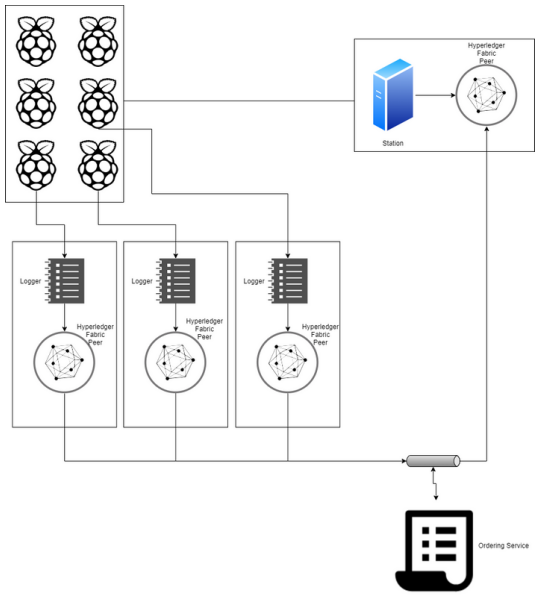


Figure 21: Hyperledger-based implementation. There is a peer for each AV (epuck and its raspberry Pi) ; the peer acts on behalf of the AV to register data in the blockchain.

6.6 Conclusion

This section presents an extension to our DDDAS system to make it more resilient to integrity problems. While the previous sections address issues related to integrity of data generated by potentially compromised sensors during one deployment, this section presents an extension to address integrity issues across several deployments. To meet this second goal, we used SDN and blockchain to store data resulting of every DDDAS deployment, and consolidate and analyze data of multiple runs guaranteeing integrity of stored data.

We analyzed available blockchain tools and selected Hyperledger because of its flexibility. We also identified issues regarding the process that devices with constrained resources must follow to register data in a given blockchain, thus we defined an alternative process.

As future work, we must run the extended system to analyze performance, integrity guarantees of stored data and detected problems.

7 Conclusions

A model-based detection and reconfiguration strategy has been implemented to decrease the attack effects on a multiple robot system. The detection has been performed with an extended Kalman filter and the non-parametric cumulative sum. Redundant sensors for measurement the physical variables of each robot, are dynamically integrated using a convex combination to increase the resilience of the system. The strategy has been simulated and implemented in a formation of six robots with one leader and two different communication topologies. Attacks with different characteristics on the leader robot and multiple robots have been deployed. Our strategy has shown the capacity to decrease the effects of an attack similar to a fault, and more sophisticated attacks such as stealthy attacks.

Additionally, we have proposed the discrete-time distributed Smith dynamics, and we have provided sufficient conditions for stability in practical implementations where computations are necessarily discrete. Moreover, we have illustrated the application of the proposed method on a real robotic platform with six e-puck v2 robots under a leader-follower scheme.

We have also addressed the problem of attacks on the communication among agents using techniques such as software defined networks (SDN). We have developed a first approach of a strategy that mitigates attacks when the information sent by one agent is corrupted.

Finally, we added an extension to the previously developed system to address integrity issues across several deployments. The extension takes advantage of blockchain and SDN technologies; the former allows us to add a tamperproof log of system behavior and the latter allows us to add an application to analyze data. Additionally, we identified and addressed some issues when connecting devices with constrained resources to a blockchain. As future work, we must further analyze performance and other issues that might arise.

References

[1] N. Quijano and K. M. Passino, "Honey bee social foraging algorithms for resource allocation: Theory and application," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 6, pp. 845–861, 2010.

[2] R. Olfati-Saber and J. S. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 6698–6703.

[3] J. Giraldo, E. Mojica-Nava, and N. Quijano, "Synchronization of isolated microgrids with a communication infrastructure using energy storage systems," *International Journal of Electrical Power & Energy Systems*, vol. 63, pp. 71–82, 2014.

[4] V. Gazi and K. M. Passino, "Stability analysis of swarms," *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 692–697, 2003.

[5] N. Chopra and M. W. Spong, "On exponential synchronization of kuramoto oscillators," *IEEE transactions on Automatic Control*, vol. 54, no. 2, pp. 353–357, 2009.

[6] J. Barreiro-Gomez, G. Obando, G. Riaño-Briceño, N. Quijano, and C. Ocampo-Martinez, "Decentralized control for urban drainage systems via population dynamics: Bogotá case study," in *Proceedings of the 2015 European Control Conference (ECC)*, 2015, pp. 2426–2431.

[7] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE transactions on intelligent transportation systems*, vol. 18, no. 11, pp. 2898–2915, 2017.

[8] P. Shakarian, J. Shakarian, and A. Ruef, "Attacking iranian nuclear facilities: Stuxnet," *Introduction to cyber-warfare: A multidisciplinary approach*, pp. 223–239, 2013.

[9] J. McLurkin, J. Rykowski, M. John, Q. Kaseman, and A. J. Lynch, "Using multi-robot systems for engineering education: Teaching and outreach with large numbers of an advanced, low-cost robot," *IEEE transactions on education*, vol. 56, no. 1, pp. 24–33, 2012.

[10] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[11] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Xinyan, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

[12] P. Guo, H. Kim, L. Guan, M. Zhu, and P. Liu, "Vcids: Collaborative intrusion detection of sensor and actuator attacks on connected vehicles," in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, 2017, pp. 377–396.

[13] A. Tiwari, B. Dutertre, D. Jovanović, T. de Candia, P. D. Lincoln, J. Rushby, D. Sadigh, and S. Seshia, "Safety envelope for security," in *Proceedings of the 3rd international conference on High confidence networked systems*, 2014, pp. 85–94.

[14] G. Sabaliauskaite, G. S. Ng, J. Ruths, and A. Mathur, "Experimental evaluation of stealthy attack detection in a robot," in *Proceedings of the Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on*, 2015, pp. 70–79.

[15] N. Bezzo, J. Weimer, M. Pajic, O. Sokolsky, G. J. Pappas, and I. Lee, "Attack resilient state estimation for autonomous robotic systems," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3692–3698.

[16] J. Giraldo and A. A. Cardenas, "Moving target defense for attack mitigation in multi-vehicle systems," in *Proactive and Dynamic Network Defense*. Springer, 2019, pp. 163–190.

[17] J. Giraldo, A. Cardenas, and R. Sanfelice, "A moving target defense to detect stealthy attacks in cyber-physical systems," in *Proceedings of the 2019 American Control Conference (ACC)*, 2019, pp. 391–396.

[18] L. Negash, S.-H. Kim, and H.-L. Choi, "Distributed observes for cyberattack detection and isolation in formation-flying unmanned aerial vehicles," *Journal of Aerospace Information Systems*, pp. 551–565, 2017.

[19] S.-H. Kim, L. Negash, and H.-L. Choi, "Cubature kalman filter based fault detection and isolation for formation control of multi-uavs," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 63–68, 2016.

[20] J.-B. Pomet, B. Thuilot, G. Bastin, and G. Campion, "A hybrid strategy for the feedback stabilization of nonholonomic mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992, pp. 129–134.

[21] J. R. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE transactions on robotics and automation*, vol. 19, no. 6, pp. 933–941, 2003.

[22] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM symposium on information, computer and communications security*, 2011, pp. 355–366.

[23] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1092–1105.

[24] R. Goebel, R. G. Sanfelice, and A. R. Teel, "Hybrid dynamical systems," *IEEE control systems magazine*, vol. 29, no. 2, pp. 28–93, 2009.

[25] ——, "Hybrid dynamical systems: modeling stability, and robustness," 2012.

[26] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*, no. LIS-CONF-2009-004, 2009, pp. 59–65.

[27] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[28] H. Xiao and C. L. P. Chen, "Leader-follower consensus multi-robot formation control using neurodynamic-optimization-based nonlinear model predictive control," *IEEE Access*, vol. 7, pp. 43 581–43 590, 2019.

[29] Z. Miao, Y. Liu, Y. Wang, G. Yi, and R. Fierro, "Distributed estimation and control for leader-following formations of nonholonomic mobile robots," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1946–1954, Oct 2018.

[30] J. Barreiro-Gomez, I. Mas, C. Ocampo-Martinez, R. Sanchez-Peña, and N. Quijano, "Distributed formation control of multiple unmanned aerial vehicles over time-varying graphs using population games," in *Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 5245–5250.

[31] N. Li and J. R. Marden, "Designing games for distributed optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 230–242, April 2013.

[32] N. Quijano, C. Ocampo-Martinez, J. Barreiro-Gomez, G. Obando, A. Pantoja, and E. Mojica-Nava, "The role of population games and evolutionary dynamics in distributed control systems: The advantages of evolutionary game theory," *IEEE Control Systems Magazine*, vol. 37, no. 1, pp. 70–97, Feb 2017.

[33] B. Liu, Y. Chen, A. Hadiks, E. Blasch, A. Aved, D. Shen, and G. Chen, "Information fusion in a cloud computing era: A systems-level perspective," *IEEE Aerospace and Electronic Systems Magazine*, vol. 29, no. 10, pp. 16–24, Oct 2014.

[34] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[35] W. H. Sandholm, *Population games and evolutionary dynamics*. MIT press, 2010.

[36] J. Barreiro-Gomez, G. Obando, and N. Quijano, "Distributed population dynamics: Optimization and control applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 2, pp. 304–314, Feb 2017.

[37] [Online]. Available: waymo.com

[38] [Online]. Available: zoox.com

[39] R. Ch, G. Srivastava, T. Reddy Gadekallu, P. K. R. Maddikunta, and S. Bhattacharya, "Security and privacy of uav data using blockchain technology," *Journal of Information Security and Applications*, vol. 55, p. 102670, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221421262030822X

[40] [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[41] P. Sun, R. Mahajan, J. Rexford, L. Yuan, M. Zhang, and A. Arefin, "A Network-state Management Service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 563–574.

[42] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-Defined Networking for Smart Grid Resilience: Opportunities and Challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015, pp. 61–68.

[43] A. Aydeger, K. Akkaya, and A. S. Uluagac, "Sdn-based resilience for smart grid communications," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Nov 2015, pp. 31–33.

[44] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 18, pp. 2713–2737, 2016.

[45] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, V. Mancuso, M. R. Sama, P. Seite, and S. Shanmugalingam, "An sdn-based network architecture for extremely dense wireless networks," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.

[46] A. Murillo, V. Gaur, J. Giraldo, A. Cardenas, and S. Rueda, "Virtual incident response functions in control systems," *Computer Networks*, vol. 135, pp. 147–159, 2018.

[47] ——, "Leveraging software-defined networking for incident response in industrial control systems," *IEEE Software*, vol. 35, pp. 44–50, 2018.