Delft University of Technology

Safe Distributed Control of Multi-Robot Systems With Communication Delays

Ballotta, Luca; Talak, Rajat

Citation (APA)
Ballotta, L., & Talak, R. (2025). Safe Distributed Control of Multi-Robot Systems With Communication Delays. *IEEE Transactions on Vehicular Technology*, *74*(7), 10137-10150. https://doi.org/10.1109/TVT.2025.3546857

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Safe Distributed Control of Multi-Robot Systems With Communication Delays

Luca Ballotta and Rajat Talak , *Member, IEEE*

Abstract—Safe operation of multi-robot systems is critical, especially in communication-degraded environments such as underwater for seabed mapping, underground caves for navigation, and in extraterrestrial missions for assembly and construction. We address safety of networked autonomous systems where the information exchanged between robots incurs communication delays. We formalize a notion of *distributed control barrier function* for multi-robot systems, a safety certificate amenable to a distributed implementation, which provides formal ground to using graph neural networks to learn safe distributed controllers. Further, we observe that learning a distributed controller ignoring delays can severely degrade safety. We finally propose a predictor-based framework to train a safe distributed controller under communication delays, where the current state of nearby robots is predicted from received data and age-of-information. Numerical experiments on multi-robot collision avoidance show that our predictor-based approach can significantly improve the safety of a learned distributed controller under communication delays.

Index Terms—Communication delays, distributed control barrier function, graph neural network, multi-robot system, safety.

I. INTRODUCTION

MOBILE autonomous robot networked systems are increasingly being conceived to aid humans in oceanbed mapping, underground subterranean navigation, search-and-rescue missions, and space exploration [1], [2], [3], [4], [5]. Safe and coordinated operation of multi-robot systems is critical to their successful deployment. However, operation environments often induce severe communication outages that make information exchange between robots imperfect and delayed [2], [6], [7]. Moreover, the safety requirements of such networked autonomous system depend on of all robots, and not just one (*e.g.,* robots need to avoid collisions with each other or move in a formation, without breaking connectivity). Therefore, there is a need to address the question of safety in conjunction with imperfect information-exchange between robots in a networked autonomous system.

Fig. 1. We propose a control strategy that keeps an autonomous multi-robot system safe via inter-robot communication. In this example, the controllers π_{safe}^i and π_{safe}^j of drones i and j use data sent by the other drone to avoid collisions while reaching the goals.
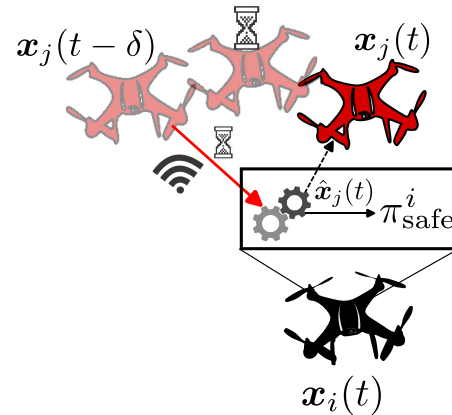
Fig. 2. Safe control via wireless communication is crucially affected by transmission delays. We propose a predictor that uses delayed measurement $\boldsymbol{x}_j(t-\delta)$ to compute the estimate $\hat{\boldsymbol{x}}_j(t)$ that is used by the controller π_{safe}^i of robot i for safe real-time operation.

A recent trend has seen control barrier functions (CBFs) as a promising tool that ensures safety of control actions by design [8]. While centralized CBF-based control theoretically ensures safety, its use is impractical for networked autonomous systems where decentralized (no communication) or distributed (inter-robot communication) control is preferred. Previous work has addressed decentralized and distributed safe control with handcrafted CBFs [9], which show promising results towards network scalability. However, computing a valid CBF for multi-robot and networked systems may be computationally hard, and

safety degradation of CBF-based control under communication delays has received little attention so far.

Contribution: We first propose a formal framework for CBF-based safe distributed control. We give a tailored definition of safe set for networked autonomous systems in Section III and formally characterize a *distributed control barrier function* that guarantees safety under local communication in Section IV-A. This theoretically justifies a distributed mechanism for safety certification and a *safe distributed controller* (Fig. 1). Our framework extends and complements previous work [10] that provides a pairwise decoupling condition on the CBF for a safe decentralized controller with backup control.

Secondly, we present a learning-based approach that builds on our formal results and leverages graph neural networks (GNNs) in Section IV-B. We consider a distributed controller where each robot computes its own control inputs based on data received from neighbors. This allows for flexible controllers that can scale to larger robot teams and require less computation and communication than distributed optimization.

Our final contribution in Section V is a novel predictor-based safe distributed control framework under communication delays. We consider delays that affect information exchange between robots, differently from previous work on CBF that addresses actuation delays [11] or tackles this issue via heuristics [7]. Indeed, communication delays are more challenging because the receiving robot must estimate the states of other robots without knowing their past inputs, while input delays can be compensated through the robot's dynamics. We propose a prediction-based framework that is agnostic to the safety task to handle delays. In it, a predictor implemented at each robot predicts the current states of nearby robots given previous state information received from them and the associated Age-of-Information [12], which captures *staleness* of received data (Fig. 2). We show that a simple alternating strategy, which alternatively learns a predictor model (given controller) and a controller model (given predictor), works.

Our experiments in Section VI show that the GNN-based controller can safely navigate a team of mobile robots to avoid collisions under perfect communication, and that the predictor-based controller significantly improves safety in the realistic case when transmissions incur communication delays.

To the best of our knowledge, this is the first work that formalizes a general distributed control barrier function suited to safe control based on inter-agent communication. Also, it is the first that proposes a general CBF-based control architecture to tackle communication delays between agents.

A. Related Work

Certified Control: Recent years have seen a surge in attention towards *control certificates*, especially control barrier functions (CBF) to meet hard constraints in the state space, often interpreted as *safety*. Control Barrier Functions were introduced in [8] inspired by barrier functions in optimization. Follow-up work extended them to uncertain dynamic environments [13], [14], online adaptation [15], and observer-controller co-design [16], to make a few examples. Decentralized collision avoidance via CBF was studied in [9], [10] without communication between

robots. Previously, a similar effort was made in [17] for leader-follower distributed control where ad-hoc barrier functions were proposed for several control objectives. Other relevant works are [18], where a resilient algorithm was proposed to tame adversarial agents, and [19], that addresses connectivity maintenance based on Laplacian eigenvalues.

Learning-based certificates: Besides optimization-based methods such as SoS, a recent trend to find a CBF is to use a machine learning model such as a neural network (NN). Reference [20] proposed BarrierNet, a learning-based tool that tunes parameters to improve performance while guaranteeing safety. Paper [21] uses an ad-hoc design to learn Lyapunov functions with guaranteed positive semi-definiteness. Works [22], [23] introduced FOSSIL, which validates the learned CBF via Satisfiability Modulo Theory (SMT). While this approach is general, the formal verification is computationally intractable for large model or state dimension. Works [24], [25] studied learnable controllers with formal guarantees implicitly provided by the considered class of closed-loop systems. The authors in [26] proposed a learning-based approach with formal verification based on linear approximations of neural networks for stochastic closed-loop dynamics.

Control with Delays: In distributed control, wireless communication affects feedback information in terms of noise and delays. These issues were tackled in [27], [28], [29], [30], [31], to name a few. Delays in optimal control design are challenging because even fairly realistic models induce nonconvex or intractable optimization problems. The communication community has proposed Age-of-Information [12], with a large body of works addressing information freshness [32], [33] or control-oriented metrics [34], [35], [36], [37]. In these works, the design focuses on scheduling of updates. Also, CBF-related research has addressed delays. Input delays were counterbalanced in [11], [13] with a predictor. Delays affecting input, dynamics, and CBF were addressed in [38], with Control Barrier Functionals for retarded systems. However, knowledge of the state at the current time is assumed, whereas realistic communication delays cause each robot to know only past states of other robots. A heuristic CBF-based control was proposed in [7] for connectivity maintenance and collision avoidance for single integrator dynamics and all-to-all communication.

Graph Neural Networks for Control: GNNs are general neural network models that inherently learn distributed architectures [39], [40], [41]. Early works [42], [43] focused on applicability of Graph Convolutional Networks to distributed control. Follow-up research [44] proposed more general models, paper [45] further broadened the framework including imitation and transfer learning, and [46] presented a physics-informed distributed controller for port-Hamiltonian systems. However, the works above do not address safety based on CBFs and neglect delays in information exchange.

B. Notation

Normal font, boldface (*e.g.,* $\boldsymbol{x}$), and calligraphic letters (*e.g.,* $\mathcal{S}$) denote scalars, vectors, and sets, respectively. The symbol $\ominus$ denotes difference between two elements in an Euclidean or

non-Euclidean space. The symbol $\odot$ denotes a permutation-invariant operation over the elements of a finite set.

II. BACKGROUND

This section reviews the main tools used in our framework.

A. Control Barrier Functions

The state of a controlled dynamical system evolves as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{1}$$

where $\boldsymbol{x}(t) \in \mathcal{X}$ denotes the system state at time t, $\boldsymbol{u}(t) \in \mathcal{U}$ is the control action at time t, and $\boldsymbol{f}$ is a locally Lipschitz function. Being us concerned with control design, the control input $\boldsymbol{u}_t(t)$ is computed by a controller π. In the following, when there is no risk of ambiguity, we occasionally omit the dependence on time t.

The system is *safe* whenever the state lies in the *safe set* $\mathcal{S} \subset \mathcal{X}$, *i.e.*, $\boldsymbol{x} \in \mathcal{S}$. The safe set is task dependent and describes safety in the state space. For example, if safety means avoiding collisions, the set $\mathcal{S}$ contains all collision-free configurations. Under this setting, the goal is to develop a certification mechanism that can attest whether the controlled dynamical system, starting from the safe set, remains in it forever. If this happens when the state $\boldsymbol{x}$ obeys dynamics (1), we say that $\mathcal{S}$ is *forward invariant* w.r.t. (1). The set $\mathcal{S}$ is typically assumed compact. The unsafe set is $\mathcal{S}_u \doteq \mathcal{X} \setminus \mathcal{S}$.

Control barrier functions (CBFs) provide an elegant way to certify whether the controlled dynamical system will remain in the safe state forever, or not. In this section, we briefly recap the main results on CBF-based safety guarantees and safe control. The interested reader is referred to [8] for details.

Assume that the safe set can be described as the superlevel set of a continuous function h, as follows:

$$\mathcal{S} = \{\boldsymbol{x} \in \mathcal{X} : h(\boldsymbol{x}) \geq 0\}. \tag{2}$$

Definition 1 (Control Barrier Function [8]): Given a set $\mathcal{S}$ defined above for a continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$, the function h is a *Control Barrier Function* (CBF) for system (1) on $\mathcal{M} \subset \mathcal{X}$ with $\mathcal{S} \subseteq \mathcal{M}$ if there exists an extended class $\mathcal{K}$-infinity function such that

$$\sup_{\boldsymbol{u} \in \mathcal{U}} \dot{h}(\boldsymbol{x}, \boldsymbol{u}) \geq -\alpha(h(\boldsymbol{x})) \quad \forall \boldsymbol{x} \in \mathcal{M}. \tag{3}$$

In this case, the *safe control set* associated with state $\boldsymbol{x}$ is

$$\mathcal{U}_{\mathcal{S}}(\boldsymbol{x}) \doteq \left\{ \boldsymbol{u} \in \mathcal{U} : \dot{h}(\boldsymbol{x}, \boldsymbol{u}) \geq -\alpha(h(\boldsymbol{x})) \right\}. \tag{4}$$

Theorem 2 (Safe control [8]): If h is a CBF for (1), any Lipschitz continuous controller $\pi : \mathcal{M} \to \mathcal{U}$ s.t. $\pi(\boldsymbol{x}) \in \mathcal{U}_{\mathcal{S}}(\boldsymbol{x})$ makes the safe state $\mathcal{S}$ forward invariant w.r.t. (1).

Remark 3 (Time derivative of CBF): The time derivative of the CBF $h(\boldsymbol{x})$, $\dot{h}(\boldsymbol{x}, \boldsymbol{u})$, is given by

$$\dot{h}(\boldsymbol{x}, \boldsymbol{u}) = L_{\boldsymbol{f}} h(\boldsymbol{x}, \boldsymbol{u}) = \nabla h(\boldsymbol{x})^{\top} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{5}$$

where $L_{\boldsymbol{f}} h$ denotes the Lie derivative of h with respect to $\boldsymbol{f}$.

Remark 4 (CBF with distributed communication): Crucially, while the CBF $h(\boldsymbol{x})$ depends on the state $\boldsymbol{x}$, its time derivative

$\dot{h}(\boldsymbol{x}, \boldsymbol{u})$ depends on both state $\boldsymbol{x}$ and control action $\boldsymbol{u}$ through the state dynamics (1). In a distributed communication context, if a unique CBF is used for the whole multi-agent system, this means that all agents (robots) should instantaneously know the states of all other agents to compute $h(\boldsymbol{x})$, and both states and control actions of all other agents to compute $\dot{h}(\boldsymbol{x}, \boldsymbol{u})$. In practice, this may be an infeasible requirement. In Section IV, we build a learning-based *distributed CBF* framework that allows robots to *locally* coordinate in order to achieve safety.

Implementing a Safe Controller: Given a CBF h, Theorem 2 offers a way to compute the control input $\boldsymbol{u}$ in (1) so as to keep the state inside the set $\mathcal{S}$. A common situation is where a nominal controller π_{nom} is already available and designed to optimize a task of interest, such as trajectory tracking or goal reaching, without guaranteeing safety. We denote nominal control inputs computed by π_{nom} as $\bar{\boldsymbol{u}}$. In this case, one can use the CBF as a filter to design a new control policy that ensures safety while being minimally deviant from the nominal controller. This can be achieved by solving the following optimization problem at each time t:

$$\boldsymbol{u}_t(t) = \min_{\boldsymbol{u} \in \mathcal{U}} \|\boldsymbol{u} - \bar{\boldsymbol{u}}(t)\|^2 \tag{6a}$$

$$\text{s.t.} \quad \dot{h}(\boldsymbol{x}(t), \boldsymbol{u}) \geq -\alpha(h(\boldsymbol{x}(t))). \tag{6b}$$

If a nominal controller is not available, one can still use (6) with $\bar{\boldsymbol{u}}(t) \equiv 0$ to minimize control effort, an approach usually called *point-wise minimum-norm control*.

Remark 5 (Convexity): The optimization problem (6) becomes much simpler when dynamics (1) are control-affine, *i.e.,*

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) + \boldsymbol{g}(\boldsymbol{x}(t))\boldsymbol{u}_t(t), \tag{7}$$

for some locally Lipschitz $\boldsymbol{f}$ and $\boldsymbol{g}$, and the set $\mathcal{U}$ is a polytope. In this case, problem (6) is a QP and can be efficiently solved. Moreover, problem (6) remains convex if $\mathcal{U}$ is convex.

B. Learning Control Barrier Functions

Because the state-of-the-art techniques to find a CBF h that meets (2)–(3) rely on Sum-of-Squares programming [8], the computational complexity becomes intractable for large state dimension. Recent works have shown that CBFs can be computed via machine learning [23], [25], [26]. In this case, h is parametrized by a trainable model h_θ (*e.g.,* a neural network) that is trained to meet the conditions that characterize a CBF. This can be done by minimizing the loss $\mathcal{L}(\theta)$ defined as

$$\mathcal{L}(\theta) \doteq \sum_{\boldsymbol{x} \sim \mathcal{S}} [\epsilon - h_\theta(\boldsymbol{x})]^+ + \sum_{\boldsymbol{x} \sim \mathcal{S}_u} [\epsilon + h_\theta(\boldsymbol{x})]^+$$

$$+ \sum_{\boldsymbol{x} \sim \mathcal{X}} \left[\epsilon - \dot{h}_\theta(\boldsymbol{x}, \boldsymbol{u}) - \alpha(h_\theta(\boldsymbol{x}))\right]^+, \tag{8}$$

where $[\cdot]^+ \doteq \max\{\cdot, 0\}$ and the summations are computed over the states sampled during training, while control actions are selected from the set $\mathcal{U}_{\mathcal{S}}(\boldsymbol{x})$, *e.g.,* via (6). The slack hyperparameter $\epsilon > 0$ softly enforces the CBF conditions on states "close" to training samples by compactness of $\mathcal{S}$.

Recent works have also proposed to learn a safety certificate and a safe controller simultaneously. The controller is parametrized as a second learning-based model, say π_ξ, and the two models are jointly trained with the loss $\mathcal{L}(\theta, \xi)$ defined as in (8) but with the control input $\boldsymbol{u}$ replaced by the learned control $\pi_\xi(\boldsymbol{x})$. To stay "close" to a nominal controller π_{nom}, one can add the deviation term $\|\pi_\xi(\boldsymbol{x}) - \bar{\boldsymbol{u}}\|$ to the loss $\mathcal{L}(\theta, \xi)$.

C. Graph Neural Networks

Graph neural networks are machine learning models for graph-based data. Each node in the graph carries one copy of a common GNN model, which takes in input data associated with that node and with its neighbors (*node features*) plus data shared from neighbors to the node (*edge features*).

The most general implementations use so-called neighborhood aggregation or message passing. Given initial node features $f_i^{(0)}$ of node i and edge features $e_{i,j}$ from in-neighbor node j to node i, the features of node i in each layer $k = 0, \dots, K-1$ are iteratively updated as

$$f_i^{(k+1)} = \gamma^{(k)}\left(f_i^{(k)}, \odot_{j \in \mathcal{N}_i} \phi^{(k)}\left(f_i^{(k)}, f_j^{(k)}, e_{j,i}\right)\right), \quad (9)$$

where ϕ and γ are differentiable functions (usually parameterized as MLPs), $\odot$ is a differentiable, permutation-invariant aggregation function (*e.g.,* sum or max), and $\mathcal{N}_i$ collects in-neighbors of node i in the message-passing graph. The output of the GNN for node i is given by features $f_i^{(K)}$ computed in the last layer.

III. Setup

In this section, we first describe the system model including robot dynamics, information exchange between robots, and safety of the networked multi-robot system in Section III-A. Then, we state the problem about safe distributed control design in Section III-B, which is addressed in the rest of the article.

A. System Model

Robot Dynamics: We consider a multi-robot system where R robots fulfill a control task. We label each robot with index $i \in \mathcal{R} \doteq \{1, \dots, R\}$. The dynamics of robot i are given by

$$\dot{\boldsymbol{x}}_i(t) = \boldsymbol{f}(\boldsymbol{x}_i(t), \boldsymbol{u}_t(t)), \quad (10)$$

where $\boldsymbol{x}_i(t) \in \mathcal{X}$ denote the state of the robot and $\boldsymbol{u}_t(t) \in \mathcal{U}$ the control action, at time t. The state of the robot may include a physical description such as its position and velocity, among other things. We use $\boldsymbol{p}_i \in \mathbb{R}^n$ to denote the position of robot i, with $n \in \{2, 3\}$. We assume that all robots have the same dynamics $\boldsymbol{f}$ and that the state and action sets $\mathcal{X}$ and $\mathcal{U}$ are identical for all robots. Vectors $\boldsymbol{x} \in \mathcal{X}^R$ and $\boldsymbol{u} \in \mathcal{U}^R$ stack respectively states and control actions of all robots.

Information Exchange: Robots can communicate through a wireless channel. Within a *distributed control* setting, local information exchange between robots is used to achieve coordination and cooperation by computing control inputs that suitably take into account the configurations and actions of other robots. Let R_c be the communication radius. Two robots within

communication range (*i.e.,* $\|\boldsymbol{p}_i - \boldsymbol{p}_j\| \le R_c$) can exchange information, *e.g.,* their states. We use the symbol $I_j(t)$ to denote the *message* transmitted by robot j to robots within range at time t, and describe it in detail later. Under *perfect-information exchange*, a robot instantaneously receives information from all the other robots within its communication radius (*e.g.,* if $\|\boldsymbol{p}_i(t) - \boldsymbol{p}_j(t)\| \le R_c$, then robot i knows $\boldsymbol{x}_j(t)$ at time t). However, a *realistic information-exchange* means that robots can have only delayed information of other robots within communication range, due to interference and wireless channel fading. We denote all time-epochs when the information was received by i from j till time t as

$$\mathcal{T}_{i \leftarrow j}(t) \doteq \{t' \le t : \text{robot } i \text{ receives } I_j(t')\} \quad (11)$$

and the information that has been received by robot i from robot j till time t by

$$\mathcal{I}_{i \leftarrow j}(t) \doteq \{I_j(t') : t' \in \mathcal{T}_{i \leftarrow j}(t)\}. \quad (12)$$

Under perfect-information exchange, the set $\mathcal{T}_{i \leftarrow j}(t)$ in (11) corresponds to all times when the robots i and j were within the communication radius, *i.e.,*

$$\mathcal{T}_{i \leftarrow j}(t) = \{t' \le t : \|\boldsymbol{p}_i(t') - \boldsymbol{p}_j(t')\| \le R_c\}. \quad (13)$$

However, under realistic information exchange, not all information sent by robot j may be received by robot i and $\mathcal{I}_{i \leftarrow j}(t)$ contains only information that has been successfully received. In particular, if communications experience nonnegligible delays, the set $\mathcal{I}_{i \leftarrow j}(t)$ will *not* contain the most recent message $I_j(t)$ and possibly other messages transmitted before time t. This crucially affects control strategies based on distributed information exchange, which must take delays into account to counterbalance them. Note that, while we use the standard disk-based communication model [7], [17], we do not assume any model on communication delays and only assume that the available information at each robot is given by (11)–(12).

Safety of Networked Autonomous System: We assume that each robot has a built-in, high-level autonomy that drives its plans and actions. This autonomy can be abstracted out as essentially implementing a nominal controller π_{nom} that computes action $\bar{\boldsymbol{u}}_i(t)$ at time t to fulfill a task of interest (*e.g.,* trajectory tracking). However, this nominal control is not guaranteed to be *safe* with respect to the network of robots (for example, it may cause two mobile robots to collide with each other). We handle inter-robot interactions via a feedback controller π_{safe} that is in charge of safety and computes corrective control actions $\tilde{\boldsymbol{u}}_i(t)$ using feedback information received from other robots. A general characterization is

$$\tilde{\boldsymbol{u}}_i(t) = \pi_{\text{safe}}\left(\boldsymbol{x}_{i,[0,t]}, \boldsymbol{u}_{i,[0,t)}, \{\mathcal{I}_{i \leftarrow j}(t)\}_{j \in \mathcal{R} \setminus \{i\}}\right). \quad (14)$$

Notation $\boldsymbol{x}_{i,[0,t]}$ (resp., $\boldsymbol{u}_{i,[0,t)}$) refers to all states (resp., control inputs) of robot i till time t. The safety-aware control input applied in (10) is $\boldsymbol{u}_t(t) = \bar{\boldsymbol{u}}_i(t) + \tilde{\boldsymbol{u}}_i(t)$. The feedback controller π_{safe} can use previous control actions $\boldsymbol{u}_{i,[0,t)}$ *excluding* $\boldsymbol{u}_t(t)$ at time t, because this is precisely computed from (14).

We next propose a notion of safety for the networked autonomous system of robots. The system is *safe* if the state $\boldsymbol{x}$

lies in the *safe set* $\mathcal{S} \subset \mathcal{X}^R$. Importantly, this involves *all* robots in the system. We make two assumptions on the structure of this safety set $\mathcal{S}$ for a networked autonomous system, which are instrumental for the theoretical result developed in Section IV.

Assumption 6 (Safe set): There exists function ψ such that

$$\mathcal{S} = \left\{ \boldsymbol{x} \in \mathcal{X}^R : \odot_{j \in \mathcal{R}} \psi(\boldsymbol{x}_i \ominus \boldsymbol{x}_j) \geq 0 \ \forall i \in \mathcal{R} \right\}, \quad (15)$$

where $\odot_{i \in \mathcal{V}}$ is a permutation-invariant operation over set $\mathcal{V}$ and $\boldsymbol{y} \ominus \boldsymbol{z} \in \mathcal{X}$ is the difference between $\boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$.[1]

Assumption 7 (Safety threshold): There exists a continuous function $\rho : \mathbb{R}^n \to \mathbb{R}$ such that $\rho(\boldsymbol{p}_i - \boldsymbol{p}_k) \geq 0$ implies $\mathrm{sgn} \odot_{j \in \mathcal{R}} \psi(\boldsymbol{x}_i \ominus \boldsymbol{x}_j) = \mathrm{sgn} \odot_{j \in \mathcal{R} \setminus \{k\}} \psi(\boldsymbol{x}_i \ominus \boldsymbol{x}_j) \ \forall k \neq i$.

Assumption 6 means that safety can be decomposed across robots. That is, the system is safe if and only if all robots are individually safe. Moreover, for each robot, safety does not depend on other robots' *identities* but only on their *states* $\boldsymbol{x}_j$'s.

Assumption 7 states that safety of any robot is independent from those robots k for which the condition $\rho(\boldsymbol{p}_i - \boldsymbol{p}_k) \geq 0$ holds. This threshold condition has an impact on necessary and sufficient communication requirements. In fact, if the communication range between robots allows this condition to hold, then safe control can be implemented based on local information exchange, as will be clarified later in Section IV-A.

The notion of safe set given by Assumptions 6 and 7 is general enough to encompass various safety applications, as exemplified next.

Example 8 (Collision avoidance): We now show how model (15) can be used to describe collision avoidance in a team of mobile robots $\mathcal{R}$. Let d_{coll} denote the minimal distance two robots need to maintain to avoid collision. Then, the safe set for collision avoidance is given by

$$\mathcal{S} = \left\{ \boldsymbol{x} \in \mathcal{X}^R : \min_{j \in \mathcal{R} \setminus \{i\}} \|\boldsymbol{p}_i - \boldsymbol{p}_j\| \geq d_{\mathrm{coll}} \ \forall i \in \mathcal{R} \right\}. \quad (16)$$

This is an instantiation of (15) in Assumption 6 with $\psi(\boldsymbol{x}_i \ominus \boldsymbol{x}_j) \doteq \|\boldsymbol{p}_i - \boldsymbol{p}_j\| - d_{\mathrm{coll}}$ and the permutation-invariant operation $\min_{j \in \mathcal{R} \setminus \{i\}}$. Assumption 7 is satisfied with $\rho(\boldsymbol{p}_i - \boldsymbol{p}_j) \doteq \|\boldsymbol{p}_i - \boldsymbol{p}_j\| - d_{\mathrm{coll}}$, meaning that, if any two robots are enough apart, they do not collide into each other.

B. Problem Statement

We aim to design a safety certification mechanism that attests whether the networked autonomous system is safe and design a minimally invasive corrective controller π_{safe}, assuming a nominal controller π_{nom}, that ensures that the state $\boldsymbol{x}$ remains safe.

Problem 1 (Distributed safety certification): Find a distributed safety certification mechanism that can attest the safety of the networked autonomous system by using the local information at each robot i, *i.e.*, states $\boldsymbol{x}_{i,[0,t]}$, control actions $\boldsymbol{u}_{i,[0,t)}$, and communicated information $\{\mathcal{I}_{i \leftarrow j}(t)\}_{j \in \mathcal{R} \setminus \{i\}}$ at time t.

Problem 2 (Distributed safe controller): Find a distributed minimally invasive controller π_{safe} that can ensure safety of

the networked autonomous system when using the nominal controller π_{nom} by using the local information at each robot i, *i.e.*, states $\boldsymbol{x}_{i,[0,t]}$, control actions $\boldsymbol{u}_{i,[0,t)}$, and communicated information $\{\mathcal{I}_{i \leftarrow j}(t)\}_{j \in \mathcal{R} \setminus \{i\}}$ at time t.

Remark 9: While works in Section II-A provides ways to design a safe minimally deviant controller, these solutions are often either designed *ad-hoc* or centralized. The latter case requires global information (the state $\boldsymbol{x}(t)$) to be known at a single unit or at each robot. Limited communication range implies that robots cannot access the full state in real time. Therefore, we need to design distributed safety mechanism and controller, while theoretically justifying when this is sufficient.

IV. CERTIFIABLY SAFE DISTRIBUTED CONTROL

In this and the following section, we discuss the design of a distributed safety certification mechanism and a distributed controller that navigate the system to remain safe.

In this section, we first describe the design in the *perfect information-exchange* case. With instantaneous communication, we formally characterize when local information exchange between robots is theoretically sufficient to achieve safety under a distributed control strategy in Section IV-A. Drawing inspiration from this theoretical result, we next propose a learning-based approach based on graph neural networks to design a distributed controller that implicitly learns to safely coordinate with other robots in Section IV-B.

A. Theoretical Foundations With Perfect Information Exchange

We first show that, for a safe set $\mathcal{S}$ that satisfies Assumptions 6 and 7, a distributed certificate ensures safety. Such a *distributed control barrier function* uses only local information available at each robot. Let the *(safety) neighborhood* of robot i be defined as $\mathcal{N}_i \doteq \{j \in \mathcal{R} \setminus \{i\} : \rho(\boldsymbol{p}_i - \boldsymbol{p}_j) \leq 0\}$ and define the augmented vector $\boldsymbol{w}_i \in 2^{\mathcal{X}^R}$ of robot i with jth element

$$\boldsymbol{w}_{ij} = \boldsymbol{x}_i \ominus \boldsymbol{x}_j \quad \forall j \in \mathcal{N}_i. \quad (17)$$

Note that $\boldsymbol{w}_i$ is defined whenever $\mathcal{N}_i \neq \emptyset$. In words, $\boldsymbol{w}_i$ collects all relative state configurations between robot i and its safety-relevant neighbors in the set $\mathcal{N}_i$. In fact, by Assumption 7, all other robots outside $\mathcal{N}_i$ are irrelevant for safety of robot i. Vector $\boldsymbol{w}_i$ will be used by the distributed CBF to define a dynamic condition on the evolution of robots' states that keeps them safe. We gather the control inputs of robots in the set $\mathcal{N}_i \cup \{i\}$ in the vector $\boldsymbol{u}_{\mathcal{N}_i}$, with jth element $\boldsymbol{u}$ for all $j \in \mathcal{N}_i \cup \{i\}$. By the chain rule, for a piece-wise differentiable function h, it holds

$$\dot{h}(\boldsymbol{w}_i, \boldsymbol{u}_{\mathcal{N}_i}) = \sum_{j \in \mathcal{N}_i} \nabla_{\boldsymbol{w}_{ij}} h(\boldsymbol{w}_i)^\top \dot{\boldsymbol{w}}_{ij} \quad (18)$$

where $\dot{\boldsymbol{w}}_{ij} = f(\boldsymbol{x}_i, \boldsymbol{u}) - f(\boldsymbol{x}_j, \boldsymbol{u})$ from (17) and (10).

Definition 10 (Distributed CBF): Let $\mathcal{M} \subset 2^{\mathcal{X}^R}$ such that $\boldsymbol{x} \in \mathcal{S}$ implies $\boldsymbol{w}_i \in \mathcal{M} \ \forall i \in \mathcal{R}$. A function $h : \mathcal{M} \to \mathbb{R}$ is a *distributed control barrier function* for (10), $i \in \mathcal{R}$, if h is continuously differentiable w.r.t. $\boldsymbol{w}_i$ for any fixed $\mathcal{N}_i$ and there exists an extended class $\mathcal{K}$ function α such that:

---

[1]In general, $\mathcal{X}$ need not be Euclidean (*e.g.*, it can be a manifold). If $\mathcal{X}$ is Euclidean, then $\ominus$ reduces to the standard difference between two vectors.

1) $h(\boldsymbol{w}_i) \geq 0$ if and only if $\odot_{j \in \mathcal{N}_i} \psi(\boldsymbol{x}_i \ominus \boldsymbol{x}_j) \geq 0$;
2) $\sup_{\boldsymbol{u}_\mathcal{N}} \dot{h}(\boldsymbol{w}_i, \boldsymbol{u}_\mathcal{N}) \geq -\alpha(h(\boldsymbol{w}_i)) \; \forall i \in \mathcal{R}, \boldsymbol{w}_i \in \mathcal{M}$.

In this case, the *safe control set* associated with $\mathcal{N}_i \cup \{i\}$ is

$$\mathcal{U}_\mathcal{S}(\boldsymbol{w}_i) \doteq \left\{ \boldsymbol{u}_\mathcal{N} \in \mathcal{U}^{|\mathcal{N}_i|+1} : \dot{h}(\boldsymbol{w}_i, \boldsymbol{u}_\mathcal{N}) \geq -\alpha(h(\boldsymbol{w}_i)) \right\}. \quad (19)$$

Theorem 11 (Distributed safety certification): The safe set (15) is equivalent to

$$\mathcal{S} = \left\{ \boldsymbol{x} \in \mathcal{X}^R : \odot_{j \in \mathcal{N}_i} \psi(\boldsymbol{x}_i \ominus \boldsymbol{x}_j) \geq 0 \; \forall i \in \mathcal{R} \right\}. \quad (20)$$

Moreover, let h be a distributed CBF for (10) for all $i \in \mathcal{R}$. Then, any Lipschitz distributed controller $\pi : 2^{\mathcal{X}^R} \to \mathcal{U}$ s.t. $\pi(\boldsymbol{w}_i) \in \mathcal{U}_\mathcal{S}(\boldsymbol{w}_i)$ makes set $\mathcal{S}$ forward invariant w.r.t. (10).

Proof: See Appendix A. □

In words, Definition 10 means that safety is inherently *local*. While the networked autonomous system is safe whenever all robots are simultaneously safe according to (15), our characterization of distributed CBF in Definition 10 and Assumption 7 imply that each robot i is endowed with its individual safety that depends only on its neighbors. In particular, the time derivative $\dot{h}(\boldsymbol{w}_i, \boldsymbol{u}_{\mathcal{N}_i})$ in condition (19) depends only on states and actions of robot i and its neighbors in $\mathcal{N}_i$. This is key to enable safe distributed control, whereby coordinating with neighbors is sufficient to formally ensure *local safety* of each robot and in turn *global safety* of the networked system according to Theorem 11. On the other hand, if at least one robot needed global system knowledge to preserve safety, a distributed control strategy based on local communication may not work.

Given a distributed CBF h, the constraint (19) can theoretically be used within a distributed optimization algorithm to compute safe control actions $\boldsymbol{u}$ for all robots i. However, this approach raises two practical issues. On the one hand, the resulting distributed optimization problem need not be convex, and convergence is not guaranteed. On the other hand, communication delays may disrupt applicability of this approach for online operation since several communication rounds and are typically needed to converge to a feasible solution. These limitations motivate us to explore a learning-based approach in the next section, whereby each robot collects information received from nearby robots but does not further communicate when computing the control actions.

Remark 12 (Communication and safety): In real applications, robot i needs to communicate at least with all robots in $\mathcal{N}_i$ to implement a distributed controller based on Theorem 11. Formally, this requires $\rho(\boldsymbol{p}_i - \boldsymbol{p}_j) \leq 0 \Rightarrow \|\boldsymbol{p}_i - \boldsymbol{p}_j\| \leq R_c$. In Example 8 this is equivalent to $R_c \geq d_{\text{coll}}$, which simply means that robots must communicate with neighbors (at least) before they collide to remain safe.

B. Learning a Safe Distributed Controller

Theorem 11 provides formal ground for the design of a distributed safety certification mechanism and controller. We identify two main benefits of a learning-based design. Firstly, it circumvents the need to find a CBF via optimization-based parametric methods, which can be computationally intractable. Secondly, it can allow robots to implicitly coordinate their

actions by just processing the received information in $\mathcal{I}_{i \leftarrow j}$, without resorting to an iterative distributed algorithm.

Assumption 6 and Definition 10 imply that the distributed CBF is *permutation-invariant*, meaning that any two neighbors of robot i can be swapped without changing the value $h(\boldsymbol{w}_i)$. Moreover, recall that we assume all robots share the same dynamics and state and control sets. Therefore, embedding the information exchanged between robots into edge features, we can model the distributed CBF h as a GNN h_θ that operates on the graph $\mathcal{G}(t) = (\mathcal{R}, \mathcal{E}(t))$, where the directed edge $(i, j) \in \mathcal{E}(t)$ if $\mathcal{I}_{i \leftarrow j}(t)$ is nonempty, *i.e.*, if robot i has knowledge of robot j at time t. Under perfect information exchange, this corresponds to robots i and j being within communication range at time t, and hence we set $\mathcal{I}_{i \leftarrow j}(t) = I_j(t) = \{\boldsymbol{x}_j(t)\}$ if $\|\boldsymbol{p}_i(t) - \boldsymbol{p}_j(t)\| \leq R_c$, as this fully describes the current state of neighbor j, and $\mathcal{I}_{i \leftarrow j}(t) = \emptyset$ otherwise. With this choice, the edge feature $e_{i,j}(t)$ from node (robot) j to node i in the GNN is the relative state $\boldsymbol{w}_{ij}(t)$, which robot i can compute online after receiving message $I_j(t)$ from robot j at time t. We model the corrective controller π_{safe} as a second graph neural network π_ξ operating on graph $\mathcal{G}(t)$ that computes corrective actions as

$$\tilde{\boldsymbol{u}}_i(t) = \begin{cases} \pi_\xi(\boldsymbol{w}_i(t)), & \mathcal{N}_i \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \quad (21)$$

In the following, we refer to (21) as (distributed) GNN-based controller. We jointly train the models h_θ and π_ξ using the following loss, averaged across all robots in $\mathcal{R}$,

$$\mathcal{L}(\theta, \xi) = \frac{1}{R} \sum_{i \in \mathcal{R}} \left(\mathcal{L}_{\text{CBF},i}(\theta, \xi) + \mathcal{L}_{\text{contr},i}(\xi) \right). \quad (22)$$

The loss term related to the distributed CBF conditions is

$$\begin{aligned} \mathcal{L}_{\text{CBF},i}(\theta, \xi) &\doteq w_\mathcal{S} \sum_{\boldsymbol{x}_i \sim \mathcal{S}} [-h_\theta(\boldsymbol{w}_i) + \epsilon]^+ \\ &+ w_{\mathcal{S}_u} \sum_{\boldsymbol{x}_i \sim \mathcal{S}_u} [h_\theta(\boldsymbol{w}_i) + \epsilon]^+ \\ &+ w_{\text{der}} \sum_{\boldsymbol{x}_i \sim \mathcal{X}} \left[-\dot{h}_\theta(\boldsymbol{w}_i, \boldsymbol{u}_{\mathcal{N}_i}) - \alpha(h_\theta(\boldsymbol{w}_i)) + \epsilon \right]^+, \end{aligned} \quad (23)$$

where $\boldsymbol{u}_{\mathcal{N}_i}$ in the argument of the time derivative $\dot{h}_\theta$ gathers the control inputs of robots in the neighborhood $\mathcal{N}_i \cup \{i\}$, and each of these is computed as

$$\boldsymbol{u} = \pi_\xi(\boldsymbol{w}_j) + \bar{\boldsymbol{u}}_j \quad \forall j \in \mathcal{N}_i \cup \{i\}, \quad (24)$$

The loss term that forces minimally invasive corrective control is

$$\mathcal{L}_{\text{contr},i}(\xi) \doteq w_{\text{contr}} \sum_{\boldsymbol{x}_i \sim \mathcal{X}} \|\pi_\xi(\boldsymbol{w}_i)\|. \quad (25)$$

For notation simplicity, we write $\boldsymbol{x}_i \in \mathcal{S}$ (resp. $\boldsymbol{x}_i \in \mathcal{S}_u$) in (23) if robot i is safe (resp. unsafe).[2] The coefficients $w_\mathcal{S}$, $w_{\mathcal{S}_u}$, w_{der}, and w_{contr} are training hyperparameters. The first two terms

---

[2]The safe set is defined for (the neighborhood of) each robot i according to Assumption 6 and (20).

(summations) in (23) correspond to states sampled respectively from the safe set and the unsafe set, whereas the third term enforces the dynamic condition of the distributed CBF. While this is theoretically required only for states in the safe set, we apply it to all training samples for robustness. During training, the time derivative $\dot{h}_\theta$ could be analytically computed by separately evaluating the gradient $\nabla_{\boldsymbol{w}_i} h_\theta(\boldsymbol{w}_i)$ w.r.t. input features $\boldsymbol{w}_i$ and the time derivative $\dot{\boldsymbol{w}}_i$ using (10); see (18). Alternatively, a computationally cheaper way is to numerically approximate $\dot{h}_\theta$ by a finite difference, *e.g.*,

$$\dot{h}_\theta(\boldsymbol{w}_i(t), \boldsymbol{u}_{\mathcal{N}_i}(t)) \approx \frac{h_\theta(\boldsymbol{w}_i(t+T_{\mathrm{s}})) - h_\theta(\boldsymbol{w}_i(t))}{T_{\mathrm{s}}}, \quad (26)$$

where T_{s} is a discrete time step and the state $\boldsymbol{x}_i(t + T_{\mathrm{s}})$ of robot i is generated from $\boldsymbol{x}_i(t)$ through (discretized) dynamics (10) applying the control action $\boldsymbol{u}_t(t)$ computed with the learned controller as $\boldsymbol{u}_t(t) = \bar{\boldsymbol{u}}_i(t) + \pi_\xi(\boldsymbol{w}_i(t))$. This model-free approach can also improve robustness to model uncertainty in (10). Details on the learning architecture used in our experiments are provided in Appendix B.

Remark 13 (Training samples): Instead of randomly sampling the state space, training samples can be drawn from simulated state trajectories. This helps to generate especially unsafe states that could occur during operation (*e.g.*, collisions) more easily than random samples. Moreover, sampling from trajectories allows one to approximate the time derivative of the distributed CBF directly from data, for example using (26).

V. SAFE DISTRIBUTED CONTROL WITH REALISTIC INFORMATION EXCHANGE

In the realistic information-exchange case, robot i does not have instantaneous access to neighbors' states because messages transmitted by other robots are delayed. This means that the distributed CBF and controller cannot use the actual value of $\boldsymbol{w}_i(t)$ to compute $\boldsymbol{u}_t(t)$. To tackle this, we design a *predictor* that estimates $\boldsymbol{w}_i(t)$ based on information received till time t.

To counterbalance communication delays, robots need to transmit more information compared to the perfect information-exchange case. Considering time-slotted communication with time index t, robot j sends the following message at time t:

$$I_j(t) = \{\boldsymbol{x}_j(t), \boldsymbol{u}_j(t-1), t\}, \quad (27)$$

where $\boldsymbol{u}_{t-1}(t-1)$ is the latest control input computed before time t. Note that robot j *cannot* transmit $\boldsymbol{u}_t(t)$ at time t since this is *computed* at time t. We let robots transmit previous control actions with the intuition that this helps the neighbors to "reconstruct" its trajectory, enhancing prediction quality. For a message $I_j(t')$ that has been received by robot i at time t, robot i constructs the corresponding *relative message* as

$$I_j^i(t,t') \doteq \{\boldsymbol{x}_i(t') \ominus \boldsymbol{x}_j(t'), \boldsymbol{u}_{t'-1}(t'-1) \ominus \boldsymbol{u}_{t'-1}(t'-1), t-t'\} \quad (28)$$

and the *relative dataset* at robot i referred to robot j is

$$\mathcal{I}_{i\leftarrow j}^i(t) \doteq \{I_j^i(t,t') : t' \in \mathcal{T}_{i\leftarrow j}(t)\}. \quad (29)$$

The time lag $t - t'$ in (28) is the *Age-of-Information* that represents staleness of data in $I_j^i(t,t')$. The predictor λ of robot i

maps received data as

$$\lambda : \mathcal{I}_{i\leftarrow j}^i(t) \longmapsto \widehat{\boldsymbol{w}}_{ij}(t), \quad (30)$$

where the output $\widehat{\boldsymbol{w}}_{ij}(t)$ is intended to be an estimate of $\boldsymbol{w}_{ij}(t)$. In words, robot i infers the current relative state of robot j by "projecting" past states and actions to the current time. Note that this mapping makes sense only if the relative dataset $\mathcal{I}_{i\leftarrow j}^i(t)$ is not empty, that is, if j is considered a neighbor at time t. To reduce storage requirements and bound the message-passing graph, robots discard all relative messages with age greater than a threshold $\Delta_{\max}$, so that j is removed from the neighbors of i when no data with age at most $\Delta_{\max}$ are received. The output of the predictor is fed to the distributed CBF and controller in place of the unknown $\boldsymbol{w}_{ij}(t)$. Specifically, the predictor-based controller computes corrective actions as

$$\tilde{\boldsymbol{u}}_i(t) = \begin{cases} \pi_\xi(\widehat{\boldsymbol{w}}_i(t)), & \exists j : \mathcal{I}_{i\leftarrow j}^i(t) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \quad (31)$$

We refer to (31) as predictor-based (distributed) control. The overall predictor-controller architecture for realistic-information exchange is schematized in Fig. 3. The dashed lines show that the CBF model h_θ (blue block) is used in training to enforce that the learned corrective control actions $\tilde{\boldsymbol{u}}_i(t)$ satisfy the safety conditions, but it is not deployed during operation.

A. Learning the Predictor

In view of the potentially complex dynamics (10) with the nonlinear controller π_ξ, we parametrize the predictor as a neural network λ_ζ. This also allows us to integrate the predictor learning into the same training procedure used for the GNN-based distributed CBF and controller. We train the predictor with the following loss that penalizes the relative mismatch from the output $\widehat{\boldsymbol{w}}_{ij}(t)$ to the current value $\boldsymbol{w}_{ij}(t)$:

$$\mathcal{L}_{\mathrm{pred}}(\zeta) = \sum_{i,j \in \mathcal{R}: \mathcal{I}_{i\leftarrow j}^i(t) \neq \emptyset} \frac{\|\widehat{\boldsymbol{w}}_{ij}(t) \ominus \boldsymbol{w}_{ij}(t)\|_{\mathcal{X}}}{\|\boldsymbol{w}_{ij}(t)\|_{\mathcal{X}}}. \quad (32)$$

When training both predictor model and distributed CBF and controller models in the same training session, we replace all values computed with perfect information, namely $h(\boldsymbol{w}_i(t))$ and $\pi_\xi(\boldsymbol{w}_i(t))$, with values computed from predicted information, namely $h(\widehat{\boldsymbol{w}}_i(t))$ and $\pi_\xi(\widehat{\boldsymbol{w}}_i(t))$, in the loss terms $\mathcal{L}_{\mathrm{CBF},i}$ and $\mathcal{L}_{\mathrm{contr},i}$ in (22). However, we have experimentally noted that learning accuracy improves when computing the time derivative $\dot{h}_\theta$ of the distributed CBF with actual values of $\boldsymbol{w}_i(t)$ rather than with predicted values $\widehat{\boldsymbol{w}}_i(t)$. Importantly, this can be done because the derivative $\dot{h}_\theta$ is explicitly computed only when training the distributed CBF model h_θ offline, but it is not needed during online operation, when only the controller π_ξ is deployed. Moreover, we have experimentally observed that alternatively training (updating) the predictor and, jointly, CBF and controller favors learning stability. This happens because the predictor is trained under a consistent control behavior, while CBF and controller learn corrective actions for given prediction capabilities. Conversely, jointly training all three models resulted in learning failure.
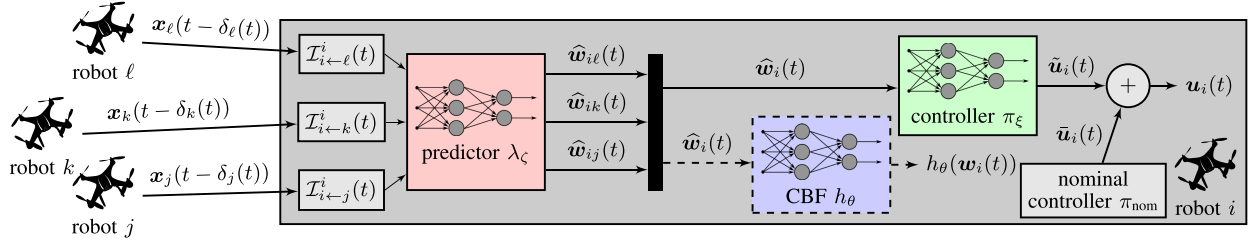
Fig. 3. Proposed distributed controller implemented on robot i. Robot i receives information from nearby robots ℓ, j, and k and computes minimally invasive control actions $u_t(t)$. Communication delays $\delta(t)$ are compensated by the predictor λ_ζ (red block). The GNN-based controller π_ξ (green block) computes corrective actions $\tilde{u}_i(t)$ to ensure safety.

Under realistic information-exchange, it makes sense to draw training samples from simulated trajectories rather than randomly from the state set $\mathcal{X}$. In fact, this allows us to train both predictor and distributed CBF and controller with states and control actions that are consistent with data received under delays. In contrast, randomly sampling states $x_i(t)$ from $\mathcal{X}$ would also require to randomly generate relative messages $I_j^i(t, t')$ that need not be consistent with feasible trajectories.

B. Heuristic Control to Enhance Performance

Learning-based controllers can hardly learn zero actions $\tilde{u}_i(t)$, which causes continuous perturbations of the nominal input $\bar{u}_i(t)$ if neighbors are present. This means that applying the learning-based controller may make the robots deviate from the desired behavior even when no corrective action is required. This effect may be counteracted by implementing task-specific adjustments. For example, in the case of point-stabilization or goal-reaching, robots may switch to the nominal controllers as soon as they are sufficiently close to the goal. While such *ad-hoc* strategies can be implemented with system-specific knowledge, in this section we explore a task-agnostic strategy that draws inspiration from the distributed CBF condition. The key intuition is that the higher the value of the CBF, the "safer" a robot is. Hence, CBF values may be thresholded to assess if corrective actions are actually needed. We propose the following switching control that corrects nominal actions according to the condition on the distributed CBF dynamics:

$$\tilde{u}_i^{\mathrm{h}}(t) = \begin{cases} 0 & \dot{h}(\hat{w}_i(t), \pi_\xi(\hat{w}_i(t))) + \alpha h(\hat{w}_i(t)) \geq \varphi\epsilon \\ \tilde{u}_i(t) & \text{otherwise.} \end{cases}$$
(33)

The parameter ϵ in (33) is the same used in training and the tunable coefficient $\varphi \geq 0$ favors safe (large φ) or aggressive behavior (small φ). In words, control (33) uses the nominal input if the next state is deemed safe by the learned CBF and applies corrective actions otherwise. Crucially, the time derivative $\dot{h}(\hat{w}_i(t), u_{\mathcal{N}_i}(t))$ cannot be computed in a distributed fashion during operation. To implement (33) in a distributed way, robots compute $\hat{w}_{ij}(t+1)$ with the predictor given available information $\mathcal{I}_{i \leftarrow j}^i(t)$, and locally estimate the time derivative as

$$\dot{h}(\hat{w}_i(t), \pi_\xi(\hat{w}_i(t))) \approx \frac{h(\hat{w}_i(t+1)) - h(\hat{w}_i(t))}{T_s}. \quad (34)$$
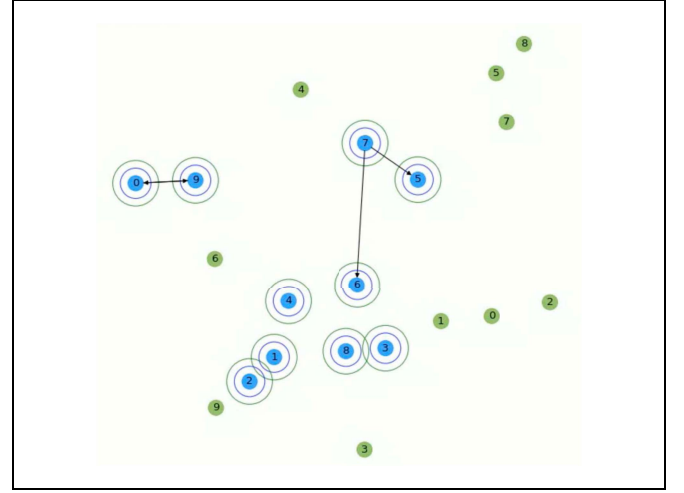

Fig. 4. Snapshot of simulation. Robots (blue) have to reach goals (green) without colliding. The inner circles around robots marks the closest distance before collision. The outer circles determine the safe set, namely robots outside them are tagged safe in training.

Remark 14 (Heuristic control): This control strategy is purely heuristic and need not retain any safety guarantee of the distributed CBF-based learned controller. We use (33)–(34) in our experiments just to compare it to the GNN-based controller, to test if we can improve performance sacrificing a little safety.

VI. EXPERIMENTS

We test our proposed controller for multi-robot navigation, where the robots have to reach goals while avoiding collisions.[3]

A. Experiment Setup

Control task: We implement and study our proposed approach for a multi-robot system concerned with collision avoidance as in Example 8. The simulation snapshot in Fig. 4 illustrates the experimental setup. We simulate mobile robots as 2D balls with radius 0.05 that move on the planar workspace $[0, 3] \times [0, 3] \subset \mathbb{R}^2$. Each robot (blue circles in Fig. 4) has to reach a pre-specified goal location (green circles) starting from an initial location,

[3]The code for training and simulations is available at https://github.com/lucaballotta/macbf-gnn.

all different across robots. To fulfill this task, each robot is equipped with a nominal controller π_{nom} that drives it from its current location to its goal through a straight line to optimize control effort, which may result in collisions. Two robots collide if the distance between their centers drops below $d_{\text{coll}} = 0.1$ (innermost ring around each robot in Fig. 4), whereas we tag safe robot states when their distance is at least 0.2 (outermost ring). The communication radius is $R_c = 1$. The arrows connecting pairs of robots in Fig. 4 show the edges $\mathcal{E}$ of the message passing graph $\mathcal{G}$ used by the distributed GNN-based controller. Several edges are missing because the snapshot is taken at the beginning of an episode ($t = 0$) and few messages $I_j(0)$ have been received because of communication delays, hence many message datasets $\mathcal{I}_{i \leftarrow j}$ are still empty.

Robot dynamics: We consider two dynamical models.

Single integrator: The state of robot i evolves as

$$\dot{\boldsymbol{x}}_i(t) = \boldsymbol{u}_t(t) \tag{35}$$

with $\boldsymbol{x}_i(t) = \boldsymbol{p}_i(t) \in \mathbb{R}^2$. Even though this model is simple, it reliably represents robot motion provided that low-level controllers track the reference velocity fast enough compared to position tracking [5], [7], [18], [47]. We set a speed limit on each direction as $\|\boldsymbol{u}\|_1 \leq 0.4$.

Dubins car: The state of robot i is a 4-dimensional vector that stacks respectively position $\boldsymbol{p}_i \in \mathbb{R}^2$, speed $v_i \in \mathbb{R}$, and heading angle $\vartheta_i \in \mathbb{R}$. Speed and heading angle are directly actuated. The dynamics are

$$\begin{bmatrix} \dot{\boldsymbol{p}}_i(t) \\ \dot{v}_i(t) \\ \dot{\vartheta}_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t)\cos(\vartheta_i(t)) \\ v_i(t)\sin(\vartheta_i(t)) \\ \boldsymbol{u}_t(t) \end{bmatrix}. \tag{36}$$

Robots are subject to acceleration limit $|u_{i,1}| \leq 10$ and angular speed limit $|u_{i,2}| \leq 100$.

At each simulation step, we discretize the dynamics via forward Euler with time step $T_s = 0.03$.

Communication model: We model communication delays as Poisson random variables with expectation that is proportional to the number of neighbors within communication range. For each robot i, transmitted message $I_i(t)$ incurs $\delta_i(t)$ delay steps, where

$$\delta_i(t) \sim \text{Pois}(c_{\text{del}}|\mathcal{N}_i(t)|), \tag{37}$$

$\mathcal{N}_i(t) \doteq \{j \in \mathcal{R} : \|\boldsymbol{p}_i(t) - \boldsymbol{p}_j(t)\| \leq R_c\}$ gathers all robots within communication range at time t, and the coefficient c_{del} depends on channel conditions and transmission power. Note that larger c_{del} or more neighbors induce longer delays. While we *generate* delays according to (37), this model is not used either to train or to test the models. Moreover, note that delays are in general asymmetric and vary overtime.

Training: We first train controllers under perfect information exchange according to Section IV-B, by manually setting to zero all transmission delays. We then train pairs of predictor and controller models as discussed in Section V. For the single integrator model we use two delay coefficients $c_{\text{del}} \in \{0.5, 0.8\}$, whereas for the Dubins car model we set $c_{\text{del}} = 1.5$. To generate training samples, we simulate trajectories of $R = 10$ robots by running multiple training episodes. We randomly assign start and goal locations to all robots at the beginning of each episode. Training episodes end either when the distance between all robots and their respective goals is at most 0.02 or if a pre-defined deadline of 500 simulation steps expires. At each simulation step, we store all robot states $\boldsymbol{x}$ and received messages $\mathcal{I}_{i \leftarrow j}$ in a buffer, from which we periodically draw a batch of training samples to update the models. The loss used for each update is the average loss across samples of the batch. To avoid imbalance in training the CBF, we sample an equal number of safe and unsafe states or, when this is not possible, we draw all unsafe states (which are usually fewer than safe states) from the buffer. We update the models every $\Delta_{\text{train}} = 512$ simulation steps for $n_{\text{desc}} = 10$ descent steps (backpropagation), For the single integrator model, we update predictor or CBF/controller 10 consecutive times before switching to the other model(s). For the Dubins car model, we first train only the controller under perfect information exchange, and then train only the predictor under communication delays. We use an ε-greedy on-policy scheme, *i.e.,* control actions applied during training are computed with the trained controller with probability $1 - \varepsilon$ and with only the nominal controller π_n with probability ε. We set a decreasing probability $\varepsilon_t = 1/t$ to favor exploration during early iterations. In the perfect information-exchange case, we train CBF and controller models for 500000 simulation steps. In the realistic information-exchange case we train CBF, controller, and predictor for a million simulation steps with single integrator dynamics, taking about 5 hours on an NVIDIA RTX A4000, while for the Dubins car model we train the predictor for 2.5 million simulation steps. Details about the learning architecture and neural network models are provided in Appendix B.

B. Results and Discussion

To evaluate safety, we compute the *safety rate* as the fraction of safe test episodes:

$$S_{\text{rate}} \doteq \frac{\# \text{ test episodes without collisions}}{\# \text{ test episodes}}. \tag{38}$$

Index S_{rate} does not account for how many collisions happen within a single episode, so it considers any episode unsafe even if a single collision occurs. For all tests, we average results over 100 Monte Carlo runs.

We first address safety under perfect information exchange, *i.e.,* instantaneous inter-robot communication, and apply our distributed GNN-based controller (21) jointly trained with the distributed control barrier function model as discussed in Section IV-B. We train all models with $R = 10$ and test how safety rate varies as we increase the number of robots deployed during testing. By doing so, we actually increase *density* of robots moving in the workspace since the latter has fixed size. The results are shown in Fig. 5. The learned controller for single integrator model easily scales to double the robot density without decreasing its safety rate, which is always 1. The Dubins' car model instead causes oscillations of the safety rate due to the more complex nonlinear behavior. Nonetheless, the safety rate remains over 80% even when doubling robot density.
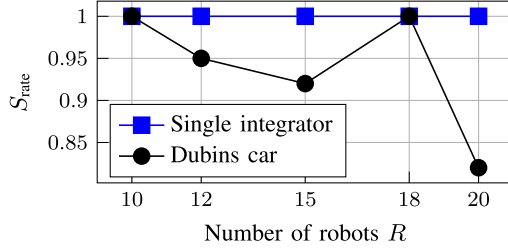
Fig. 5.   Safety rates with controller (21) in the perfect information-exchange case.
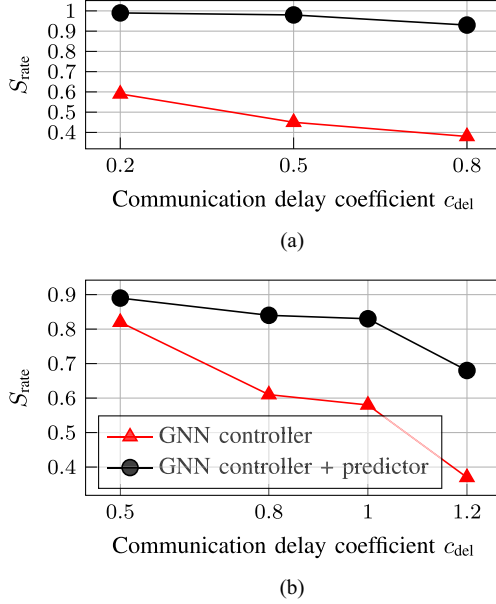

(a)

(b)

Fig. 6.   Safety rates with controller (21) ("GNN controller") and with predictor-based controller (31) ("GNN controller + predictor") in the realistic information-exchange case. Our proposed control architecture sizably improves safety under communication delays. (a) Single integrator model. (b) Dubins car model.

We then turn to the realistic information-exchange setting, where we compare the two distributed control approaches with and without predictor. Fig. 6 conveys the main message. Awareness of communication delays is crucial to effectively handle data received by nearby robots during operation. The "GNN controller" is the distributed GNN-based controller (21) trained under perfect-information exchange with no prediction. The "GNN controller + predictor" is the predictor-based controller (31) illustrated in Fig. 3 trained under realistic-information exchange, where the predictor estimates the relative states $\boldsymbol{w}_{ij}$ from delayed information $\mathcal{I}_{i \leftarrow j}^{i}$ tagged with AoI. The plot shows that the predictor-based controller sizably increases the safety rate as opposed to the control (21) that neglects communication delays. This witnesses that the CBF condition critically depends on timeliness of information, especially at the boundary between safe and unsafe regions (*i.e.,* when two robots are about to collide). To test "GNN controller + predictor" trained under realistic information-exchange for single integrator dynamics (Fig. 6(a)), we use the same delay coefficient c_{del} set in training, except for the test with delay coefficient $c_{\mathrm{del}} = 0.2$ where we use

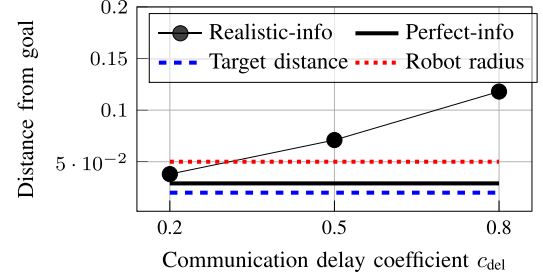Fig. 7.   Average final distance from goal with trained GNN-based controller.

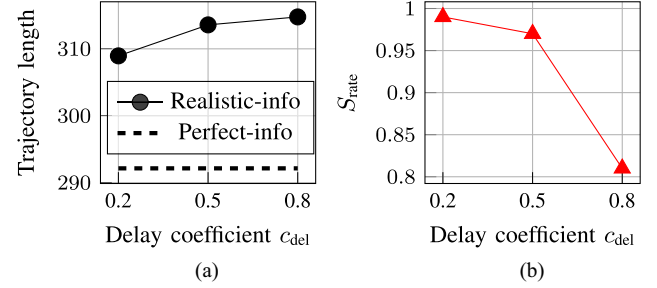(a)   (b)

Fig. 8.   Experiments with heuristic control (33) and single integrator model. (a) Average trajectory length. (b) Safety rate.

the controller trained with $c_{\mathrm{del}} = 0.5$. Conversely, the predictor for the Dubins car model is trained under a large coefficient $c_{\mathrm{del}} = 1.5$ and we study safety for smaller delays (Fig. 6(b)). The tests suggest that the controllers generalize to delay distributions with smaller delay expectations. It is important to remark that we consider *distributed controllers* because we directly deploy the trained GNN-based models π_{ξ}.

Goal-Reaching: While the GNN-based controller π_{ξ} provides a satisfactory safety level, we numerically observe that it hardly outputs zero corrective actions $\tilde{\boldsymbol{u}}_i(t)$. This causes the robots to settle off their respective goal locations. Fig. 7 shows the average robot-to-goal distance after the episode deadline expires under single-integrator model both with perfect information-exchange (solid line) and realistic information-exchange (marks). Visual inspection reveals that robots settle close to goals with a steady-state error. The latter increases with communication delays, meaning that the robots apply stronger corrections to compensate for more outdated data. The dashed blue line shows the target distance below which the goal is reached. For context, robots are simulated as circles with radius 0.05 as shown by the red dotted line.

We explore how the heuristic switching controller (33) described in Section V-B can counteract this effect. Recall that this heuristics may not retain any safety guarantee of the GNN-based distributed controller π_{ξ}. For this experiment, we deploy the controller trained with delay coefficient $c_{\mathrm{del}} = 0.5$ in all tests and set $\varphi = \{0.8, 1, 1.1\}$ in (33) to test with $c_{\mathrm{del}} = \{0.2, 0.5, 0.8\}$, respectively, to reflect the different delay distributions. Using (33), the robots almost always reach their respective goals before the deadline. The average trajectory length (number of steps) from start to goal using controller (33) are shown in Fig. 8(a). The trajectory lengths increase with delays, as expected because
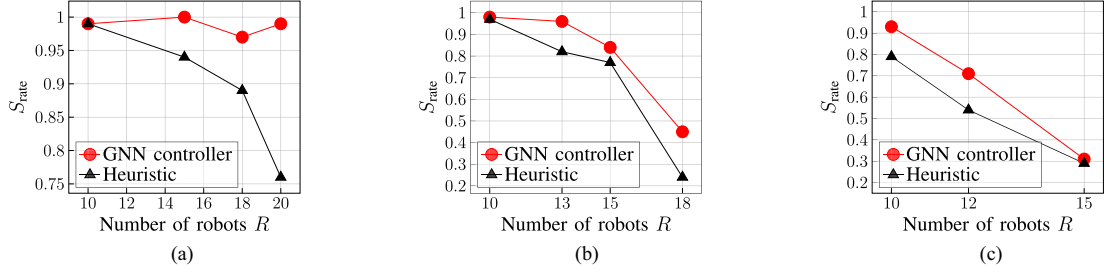
Fig. 9. Safety rate when increasing robot density in test ($R = 10$ in training). (a) Coefficient $c_{del} = 0.2$. (b) Coefficient $c_{del} = 0.5$. (c) Coefficient $c_{del} = 0.8$.
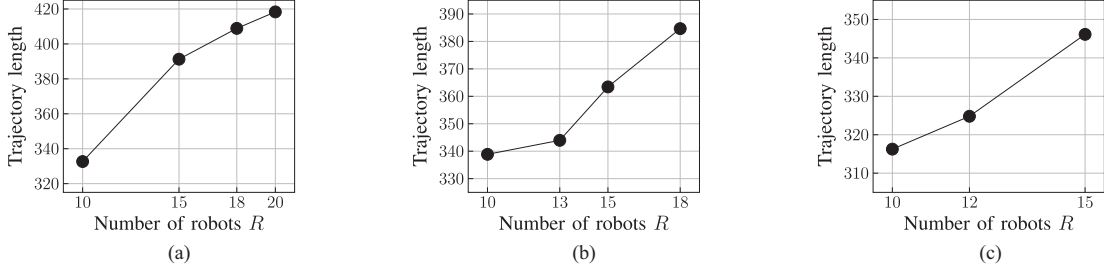

Fig. 10. Average trajectory length with heuristic control (33) when increasing robot density in test ($R = 10$ in training). (a) Coefficient $c_{del} = 0.2$. (b) Coefficient $c_{del} = 0.5$. (c) Coefficient $c_{del} = 0.8$.

robots need to act more cautiously. On the other hand, Fig. 8(b) shows that the safety rates are lower than those obtained with the trained controller and shown in Fig. 5. This is because the heuristic control (33) sacrifices safety to achieve better goal-reaching performance. In particular, the predictor introduces extra uncertainty when estimating the CBF derivative in (33), which causes safety degradation especially with long delays.

To get a sense of the performance loss due to communication delays, the dashed line in Fig. 8(a) shows the average trajectory length obtained under perfect-information exchange and by computing the CBF derivative in (33) in a centralized fashion.[4] This case is however impractical either under communication delays or if the robots do not know each other's goals (which here are sufficient to compute nominal control inputs), and we show it just for comparison as upper bound for performance.

Scalability for realistic information-exchange: Differently from the perfect-information exchange, for which we study scalability in Fig. 5, increasing the number of robots can negatively affect scalability in the presence of communication delays. In fact, by increasing *density* of robots in the workspace, model (37) generally produces longer delays because robots communicate with more neighbors. This means that increasing the number of robots also causes a distribution shift from delays observed during training.

Fig. 9 shows the safety rates obtained in test under single integrator model when increasing the robot density (*i.e.,* number of robots with fixed workspace) from training settings. We show safety rates with both the GNN-based controller (21) (label "GNN controller") and the heuristic controller (33) (label "Heuristic"). For "GNN controller", we deploy the models trained with delay coefficient $c_{del} = 0.5$ in the tests with $c_{del} =$

0.2 and with $c_{del} = 0.5$, while we deploy the models trained with $c_{del} = 0.8$ in the test with $c_{del} = 0.8$. For "Heuristic", we use only the models trained with $c_{del} = 0.5$. While the controller is able to generalize till double the robot density experienced in training when tested with $c_{del} = 0.2$, a safety degradation can be observed with the other two delay coefficients as the system is made more cluttered. This is not surprising because, while acceptable safety levels are maintained for (slightly) denser scenarios, the controller cannot generalize to arbitrarily longer delays. Moreover, we see that "Heuristic" provides smaller safety rates because it lacks statistical guarantees and pushes on performance. Therefore, this study suggests that a careful assessment of communication delays is crucial to train the predictor-controller models in an effective manner.

We also study how performance changes as robot density increases. Fig. 10 shows the average lengths of trajectories run by robots to reach their goals under the heuristic control (33), analogously to the black circles in Fig. 8(a). More cluttered environments force robots to navigate longer paths to reach their goals, which agrees with intuition.

VII. LIMITATIONS AND FUTURE RESEARCH

While our theoretical results and methodology are quite general, we have demonstrated it only for the task of collision avoidance under single integrator and Dubins car dynamic models. Showing applicability of our methodology to other tasks and dynamics is an important research avenue to be explored in future work. In particular, we have identified the prediction as a serious bottleneck for safety under realistic information-exchange for more complicated dynamics, which urges more research to apply the proposed framework to real-world applications. This could involve a different predictor architecture or embedding prediction uncertainty, such as via Gaussian Processes

---

[4]The delay coefficients on the x-axis are irrelevant for this plot since instantaneous communication is implemented.

or Stochastic Barrier Functions [26]. Alternatively, a different predictor-controller interplay may be thought so as to enhance prediction capabilities via a tighter coordination of control actions. Another point that the present work does not address is formal verification of the leaned distributed CBF, which is crucial for deployment in safety-critical applications. Finally, the promising results with the predictor-based framework open a research avenue to controlling general delay systems, and could be integrated with actuation delays in either model- or learning-based fashion.

VIII. CONCLUSION

We have proposed distributed control barrier functions for networked autonomous systems and studied a learning-based approach based on GNN models to jointly learn a distributed CBF and a safe distributed controller. We have empirically shown that communication delays can disrupt safety certification if not accounted for, and that a predictor-based approach where a predictor model is alternatively trained with the CBF and controller in the presence of delays can restore safety. In particular, using AoI of exchanged information has proven useful to handle delays. This allows for a simple, yet principled, approach to solve other control problems with delays, which are typically a challenge in controller design.

APPENDIX A
PROOF OF THEOREM 11

Characterization (20) follows directly from Assumptions 6 and 7 and the definition of $\mathcal{N}_i$. In view of Definition 10 and Assumption 7, if all control inputs are such that $\boldsymbol{u}_{\mathcal{N}_i} \in \mathcal{U}_S(\boldsymbol{w}_i) \, \forall i$ and as long as no neighborhood change, then h is differentiable and $\mathcal{S}$ is forward invariant by Theorem 2. We need to prove that the $\mathcal{S}$ is forward invariant when neighborhoods change. Assume $\rho(\boldsymbol{p}_i - \boldsymbol{p}_j) > 0$ (i.e., $j \notin \mathcal{N}_i$) at time t and $\rho(\boldsymbol{p}_i - \boldsymbol{p}_j) \leq 0$ (i.e., $j \in \mathcal{N}_i$) at time $t + \epsilon$ for some $\epsilon > 0$. Because ρ and $\boldsymbol{p}$ are continuous, there exists $\epsilon_1 \in (0, \epsilon)$ such that $\rho(\boldsymbol{p}_i - \boldsymbol{p}_j) = 0$ at time $t + \epsilon_1$. Thus, $j \in \mathcal{N}_i$ at time $t + \epsilon_1$ and, by Assumption 7, if $h(\boldsymbol{w}_i) \geq 0$ at time t and $\boldsymbol{u}_{\mathcal{N}_i} \in \mathcal{U}_S(\boldsymbol{w}_i)$ for every $t' \in [t, t + \epsilon_1)$, then $h(\boldsymbol{w}_i) \geq 0$ at time $t + \epsilon_1$. Hence, the set $\mathcal{N}_i \cup \{i\}$ with $j \in \mathcal{N}_i$ complies with $\boldsymbol{x} \in \mathcal{S}$ at time $t + \epsilon_1$ and condition (19) applied to this new neighborhood ensures $h(\boldsymbol{w}_i) \geq 0$ for $t' \geq t + \epsilon_1$.

APPENDIX B
LEARNING ARCHITECTURE USED FOR EXPERIMENTS

Predictor: The predictor λ_ζ estimates current states differences $\widehat{\boldsymbol{w}}_{ij}(t)$ between robots i and j, which are fed to the distributed CBF model h_θ and distributed controller model π_ξ. To leverage data streams received from neighbors, we parametrize the predictor as a recurrent neural network to learn correlations across message sequences in $\mathcal{I}_{i \leftarrow j}^i(t)$. We use PyTorch LSTM with four layers, hidden size 256, and dropout probability 0.1. Also, we discard all messages with AoI greater than $\Delta_{\max} = 5$ steps for the single integrator model and $\Delta_{\max} = 10$ steps for the Dubins Car model.

CBF and Controller: We design h_θ and π_ξ as compositions of one GNN layer followed by an MLP. In the perfect information-exchange setting, when current relative state information $\boldsymbol{w}_i(t)$ is readily available, we set

$$h_\theta(\boldsymbol{w}_i(t)) = m_h \left(f_i^{(K)}(t) \right) \tag{39}$$

$$\pi_\xi(\boldsymbol{w}_i(t)) = m_\pi \left(f_i^{(K)}(t), \bar{\boldsymbol{u}}_i(t) \right), \tag{40}$$

where we set $K = 1$ and the GNN model Γ extracts features at time t as (cf. (9))

$$f_i^{(k+1)}(t) = \gamma \left(f_i^{(k)}(t), \odot_{j \in \mathcal{R}: \mathcal{I}_{i \leftarrow j}(t) \neq \emptyset} \phi\left(\boldsymbol{w}_{ij}(t)\right) \right)$$

$$f_i^{(K)}(t) = \Gamma\left(\boldsymbol{w}_i(t)\right). \tag{41}$$

The models m_h and m_π are MLPs that map the extracted features to CBF values and corrective control inputs, respectively. Under realistic information-exchange, the relative states $\boldsymbol{w}_{ij}(t)$ are replaced by their estimated values $\widehat{\boldsymbol{w}}_{ij}(t)$ computed by the predictor. We also feed the reference signal $\bar{\boldsymbol{u}}_i(t)$ to the controller module m_π to enhance performance. Because we embed within the (estimated) relative states $\boldsymbol{w}_i(t)$ into the edge features between nodes, we set the initial robot features $f_i^{(0)}(t)$ simply as vector of one's for all robots.[5]

In our implementation of the GNN model Γ, we parametrize both ϕ and γ in (41) as MLPs with two hidden layers, each with 2048 nodes, and ReLu activation functions. We use an attention-type aggregation function $\odot$ because, intuitively, not all neighbors carry the same importance to safety – closer neighbors have more chance of colliding than far-away ones. Specifically, we use AttentionalAggregation [48] whereby the relevance of each neighbor is parametrized via an MLP (that we implement with two hidden layers with 128 nodes each and ReLu activation), followed by softmax weighing of the neighbors' scores. Finally, we parametrize the MLPs m_h and m_π in (40)–(39) with three hidden layers respectively with 512, 128, and 32 nodes, all with ReLu activation except for the tanh output activation of m_h that bounds the CBF values within the interval [0,1].

Given a batch with S samples, each learning iteration updates the predictor model with the batch loss

$$\mathcal{L}_{\text{pred}}^S(\zeta) = \frac{1}{S} \sum_{s=1}^{S} \mathcal{L}_{\text{pred}}(\zeta; \boldsymbol{x}^s, \mathcal{I}^s) \tag{42}$$

and the distributed CBF and controller models are jointly trained with the batch loss

$$\mathcal{L}^S(\theta, \xi) = \frac{1}{S} \sum_{s=1}^{S} \mathcal{L}(\theta, \xi; \boldsymbol{x}^s, \mathcal{I}^s), \tag{43}$$

where $\mathcal{L}_{\text{pred}}$ and $\mathcal{L}$ are respectively defined in (32) and (22), and the sth sample is given by the pair $(\boldsymbol{x}^s, \mathcal{I}^s)$ that contains respectively the state of the system and received data $\mathcal{I}_{i \leftarrow j}$ of all robots i for all neighbors j. In the loss term related to the

---

[5]While robot-specific information might be included in $f_i^{(0)}(t)$, exploring this design option is out of scope of the present contribution.

TABLE I
PARAMETERS AND HYPERPARAMETERS USED WITH SINGLE INTEGRATOR MODEL

	c_{del}	$w_{\mathcal{S}}$	$w_{\mathcal{S}_u}$	w_{der}	α	ϵ	w_{contr}
exp. 1	0.5	1	1	0.7	1	0.02	0.001
exp. 2	0.8	0.9	1	0.7	1	0.02	0.0005

TABLE II
PARAMETERS AND HYPERPARAMETERS USED WITH DUBINS CAR MODEL

$w_{\mathcal{S}}$	$w_{\mathcal{S}_u}$	w_{der}	α	ϵ	w_{contr}
1	1.2	0.5	1	0.02	0.001

distributed CBF (23), we set a linear class $\mathcal{K}$ function $\alpha(x) = \alpha x$ with slope $\alpha > 0$. Moreover, we numerically approximate the CBF derivative (that we use only in training) in (23) via (26). Parameters and learning hyperparameters used for training are reported in Tables I and II. Again, when training together predictor model and distributed CBF and distributed controller models, we replace exact values of $\boldsymbol{w}_{ij}(t)$ with their predicted values $\widehat{\boldsymbol{w}}_{ij}(t)$ when computing the distributed CFB $h_\theta(\widehat{\boldsymbol{w}}_i(t))$ and the control input $\pi_\xi(\widehat{\boldsymbol{w}}_i(t))$, but we use the ground truth (*i.e.,* actual values of $\boldsymbol{w}_i(t)$ and of $\boldsymbol{w}_i(t + T_s)$) to compute the time derivative of the distributed CBF via (26).

REFERENCES

[1] J. Hu, P. Bhowmick, and A. Lanzon, "Group coordinated control of networked mobile robots with applications to object transportation," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 8269–8274, Aug. 2021.

[2] S. Li, S. Zhang, G. He, and T. Jiang, "Discrete-time flocking control in multi-robot systems with random link failures," *IEEE Trans. Veh. Technol.*, vol. 73, no. 9, pp. 12290–12304, Sep. 2024.

[3] X. Zhou, W. Wang, T. Wang, Y. Lei, and F. Zhong, "Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11691–11703, Dec. 2019.

[4] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, Dec. 2020.

[5] D. Lee, A. Franchi, H. I. Son, C. Ha, H. H. Bülthoff, and P. R. Giordano, "Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles," *IEEE/ASME Trans. Mechatron.*, vol. 18, no. 4, pp. 1334–1345, Aug. 2013.

[6] M. Pezzutto, M. Farina, R. Carli, and L. Schenato, "Remote MPC for tracking over lossy networks," *IEEE Control Syst. Lett.*, vol. 6, pp. 1040–1045, 2022.

[7] B. Capelli, H. Fouad, G. Beltrame, and L. Sabattini, "Decentralized connectivity maintenance with time delays using control barrier functions," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 1586–1592.

[8] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[9] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 661–674, Jun. 2017.

[10] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Syst. Lett.*, vol. 5, no. 1, pp. 127–132, Jan. 2021.

[11] M. Jankovic, "Control barrier functions for constrained control of linear systems with input delay," in *Proc. IEEE Annu. Amer. Control Conf.*, 2018, pp. 3316–3321.

[12] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synth. Lectures Commun. Networks*, vol. 12, no. 2, pp. 1–224, 2019.

[13] T. G. Molnar, A. K. Kiss, A. D. Ames, and G. Orosz, "Safety-critical control with input delay in dynamic environment," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 4, pp. 1507–1520, Jul. 2023.

[14] V. Hamdipoor, N. Meskin, and C. G. Cassandras, "Safe control synthesis using environmentally robust control barrier functions," *Eur. J. Control*, vol. 74, 2023, Art. no. 100840.

[15] J. Breeden and D. Panagou, "Predictive control barrier functions for online safety critical control," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 924–931.

[16] D. R. Agrawal and D. Panagou, "Safe and robust observer-controller synthesis using control barrier functions," *IEEE Control Syst. Lett.*, vol. 7, pp. 127–132, 2023.

[17] D. Panagou, D. M. Stipanović, and P. G. Voulgaris, "Distributed coordination control for multi-robot networks using Lyapunov-like barrier functions," *IEEE Trans. Autom. Control*, vol. 61, no. 3, pp. 617–632, Mar. 2016.

[18] M. Cavorsi, B. Capelli, L. Sabattini, and S. Gil, "Multi-robot adversarial resilience using control barrier functions," in *Proc. Robot., Sci. Syst.*, 2022, vol. 18, pp. 797–815.

[19] P. Ong, B. Capelli, L. Sabattini, and J. Cortés, "Nonsmooth control barrier function design of continuous constraints for network connectivity maintenance," *Automatica*, vol. 156, 2023, Art. no. 111209.

[20] W. Xiao et al., "BarrierNet: Differentiable control barrier functions for learning of safe robot control," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 2289–2307, Jun. 2023.

[21] N. Gaby, F. Zhang, and X. Ye, "Lyapunov-net: A deep neural network architecture for Lyapunov function approximation," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 2091–2096.

[22] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of Lyapunov neural networks," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 773–778, Jul. 2021.

[23] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks," in *Proc. 24th Int. Conf. Hybrid Syst., Computation Control*, 2021, pp. 1–11.

[24] M. Zakwan, M. d'Angelo, and G. Ferrari-Trecate, "Universal approximation property of Hamiltonian deep neural networks," *IEEE Control Syst. Lett.*, vol. 7, pp. 2689–2694, 2023.

[25] L. Furieri, C. L. Galimberti, and G. Ferrari-Trecate, "Neural system level synthesis: Learning over all stabilizing policies for nonlinear systems," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 2765–2770.

[26] F. B. Mathiesen, S. C. Calvert, and L. Laurenti, "Safety certification for stochastic systems via neural barrier functions," *IEEE Control Syst. Lett.*, vol. 7, pp. 973–978, 2023.

[27] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1453–1464, Sep. 2004.

[28] U. Münz, A. Papachristodoulou, and F. Allgöwer, "Delay robustness in consensus problems," *Automatica*, vol. 46, no. 8, pp. 1252–1265, 2010.

[29] N. Matni, "Communication delay co-design in $\mathcal{H}_2$-Distributed control using atomic norm minimization," *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 2, pp. 267–278, Jun. 2017.

[30] M. A. Gomez, A. V. Egorov, S. Mondié, and W. Michiels, "Optimization of the $\mathcal{H}_2$ norm for single-delay systems, with application to control design and model approximation," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 804–811, Feb. 2019.

[31] F. Branz, R. Antonello, M. Pezzutto, S. Vitturi, F. Tramarin, and L. Schenato, "Drive-by-Wi-Fi: Model-based control over wireless at 1 kHz," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 3, pp. 1078–1089, May 2022.

[32] V. Tripathi, R. Talak, and E. Modiano, "Age optimal information gathering and dissemination on graphs," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 54–68, Jan. 2023.

[33] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 1, pp. 15–28, Feb. 2020.

[34] V. Tripathi and E. Modiano, "A whittle index approach to minimizing functions of age of information," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput.*, 2019, pp. 1160–1167.

[35] O. Ayan, M. Vilgelm, M. Klügel, S. Hirche, and W. Kellerer, "Age-of-information vs. value-of-information scheduling for cellular networked control systems," in *Proc. 10th ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2019, pp. 109–117.

[36] M. Klügel, M. H. Mamduhi, S. Hirche, and W. Kellerer, "AoI-penalty minimization for networked control systems with packet loss," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 189–196.

[37] J. P. Champati, M. H. Mamduhi, K. H. Johansson, and J. Gross, "Performance characterization using AoI in a single-loop networked control system," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2019, pp. 197–203.

[38] A. K. Kiss, T. G. Molnar, A. D. Ames, and G. Orosz, "Control barrier functionals: Safety-critical control for time delay systems," *Int. J. Robust Nonlinear Control*, vol. 33, no. 12, pp. 7282–7309, 2023.

[39] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2016. [Online]. Available: https://openreview.net/forum?id=SJU4ayYgl

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ

[41] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=F72ximsx7C1

[42] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, "Graph policy gradients for large scale robot control," in *Proc. Conf. Robot Learn.*, 2020, pp. 823–834.

[43] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11785–11792.

[44] R. Kortvelesy and A. Prorok, "ModGNN: Expert policy approximation in multi-agent systems with a modular graph neural network architecture," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9161–9167.

[45] F. Gama, Q. Li, E. Tolstaya, A. Prorok, and A. Ribeiro, "Synthesizing decentralized controllers with graph neural networks and imitation learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1932–1946, 2022.

[46] E. Sebastián, T. Duong, N. Atanasov, E. Montijano, and C. Sagüés, "LEMURS: Learning distributed multi-robot interactions," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7713–7719.

[47] E. Rossi et al., "Coordinated multi-robot trajectory tracking control over sampled communication," *Automatica*, vol. 151, 2023, Art. no. 110941.

[48] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3835–3845.

Luca Ballotta received the master's degree in automation engineering and the Ph.D. degree in information engineering from the University of Padova, Padua, Italy, in 2019 and 2023, respectively. In 2020 and 2022, he was a Visiting Student with the Massachusetts Institute of Technology, Cambridge, MA, USA. He is currently a Postdoctoral Researcher with the Delft Center for Systems and Control, Delft University of Technology, Delft, the Netherlands. His research interests include multi-agent systems and networked control under delays, resilient distributed control and distributed learning, safe control, and control under sparsity constraints. He was the recipient of the Young Author Prize at the 2020 IFAC World Congress and also the Finalist of the 2024 EECI Ph.D. Award.

Rajat Talak (Member, IEEE) received the B.Tech. degree from the Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, Mangalore, India, in 2010, the Master of Science degree from the Department of Electrical Communication Engineering, Indian Institute of Science, Bengaluru, India, in 2013, and the Ph.D. degree from the Laboratory of Information and Decision Systems, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2020. He was a Postdoctoral Associate with the Department of Aeronautics and Astronautics, MIT. He is currently a Research Scientist with the Department of Aeronautics and Astronautics, MIT. His research interests include robot perception, optimization and learning, autonomous systems, and communication networks. He was the recipient of the Best Paper Award at the ACM MobiHoc 2018 and the Gold medal for his master's thesis.