Provably Safe Multi-Robot Coordination with Unreliable Communication

Anna Mannucci[1], Lucia Pallottino[2], Federico Pecora[3]

***Abstract*—Coordination is a core problem in multi-robot systems, since it is key to ensuring safety and efficiency. Both centralized and decentralized solutions have been proposed, however, most assume perfect communication. This article proposes a centralized method which removes this assumption, and is suitable for fleets of robots driven by generic second order dynamics. We formally prove that (i) safety is guaranteed if communication errors are limited to delays, and (ii) the probability of unsafety is bounded by a function of the channel model in networks with packet loss. The approach exploits knowledge of the network's non-idealities to ensure the best possible performance of the fleet. The method is validated via several experiments with simulated robots.**

***Index Terms*—Multi-Robot Systems; Planning, Scheduling and Coordination; Formal Methods in Robotics and Automation.**

I. Introduction

MODELING and accounting for the limitations of communication is extremely important in multi-robot systems, both for performance and for safety purposes. Wireless technologies are necessary for robot mobility, however, they suffer from connectivity loss, spectrum interference and high latency handover [1]. Fleets of AGVs are particularly important in harsh environments such as underground mines, where it is notoriously problematic to guarantee reliable communication [2]. Communication standards providing low delays and packet loss have been proposed for use in automotive [3] and industrial applications [4]. However, approaches that tackle these limitations in conjunction with the multi-robot coordination problem make strong assumptions on paths and on robot kinodynamics (see, e.g., [5, 6]).

Conversely, the literature on multi-robot coordination overlooks the realities of the communication infrastructure, typically focusing on one or more of the following key requirements of real-world applications [7]: the robots in the fleet are heterogeneous and subject to non-trivial kinodynamic constraints; goals become known and are posted asynchronously (e.g., by a separate and pre-existing workflow management system); the need to discretize the environment and/or robot paths should be minimized, as this raises the cost of deployment; it should be possible to regulate the motions of robots through shared regions according to application-specific priorities; methods should scale to dozens of robots and large environments; and coordination should guarantee safety (no collisions) and liveness of the fleet (robots will eventually reach their goals).

To the best of our knowledge, no existing approach adheres to the requirements above while accounting for communication limitations. Table I summarizes how selected approaches[1] to multi-robot coordination relate to these requirements and to communication-related aspects. The specific limitations of existing approaches (see detailed notes in Table I) reveal the reason why centralized decision making structures still dominate in industrial practice: while distributed or decentralized solutions are by nature relatively robust to communication failures [9, 11–15], centralized methods deliver predictable fleet behavior, provable safety and liveness, and high performance [8, 10, 16–18].

In this paper, we explore the level of safety that can be guaranteed with knowledge of a model of the communication channel. Our algorithm builds on a supervisory control method proposed in [18], where precedences for potentially colliding pairs of robots are updated continuously, taking into account robot kinodynamics. While maintaining its good properties (see Table I), we extend [18] in the following ways: (i) the assumption of perfect communication (necessary for safety in [18]) is removed; (ii) the probability of robots not respecting a precedence is guaranteed to be below a desired threshold (knowing an upper-bound of the packet loss probability and the maximum transmission delay); (iii) we prove that this threshold is zero if network disturbances are limited to delays; (iv) we provide a set of rules to design the network to ensure a desired level of safety, avoiding network congestion. All properties are validated formally, and an experimental evaluation highlights the behavior of the algorithm in realistically-sized fleets, with typical channel models. Note that properties (iii) and (iv) provide a means to design the communication infrastructure considering the requirements of the fleet. To the best of our knowledge, this issue has never before been studied in the multi-robot research area.

II. Notation and preliminaries

We first introduce key concepts and a high-level description of the algorithm in [18] as a basis for our approach. Hence, for now, perfect communication is assumed. This assumption

Manuscript received: Feb, 24, 2019; Apr, 9, 2019; Accepted Jun, 4, 2019.
This paper was recommended for publication by Editor Nak Young Chong upon evaluation of the Associate Editor and Reviewers' comments. *This work is funded by the EU's Horizon 2020 research and innovation program under agreement no. 732737 (ILIAD), and by the Swedish Knowledge Foundation (KKS) under the Semantic Robots research profile.

[1]PhD student and [2]Associate Professor at Research Center "E. Piaggio", University of Pisa, Italy anna.mannucci@ing.unipi.it, lucia.pallottino@unipi.it
[3]Associate Professor, Center for Applied Autonomous Sensor Systems (AASS), Örebro University, Sweden federico.pecora@oru.se

Digital Object Identifier (DOI): see top of this page.

[1]A comprehensive overview of multi-robot coordination methods is beyond the scope of this paper, and the interested reader is referred to [8, 9]. We also exclude approaches for solving the task allocation and/or motion planning problem jointly with coordination, e.g., Multi-Agent Path Finding (MAPF) methods; the problem at hand in these cases is intractable [10], hence approaches tend to adhere to few of the requirements listed above.

	Centralized coordination					Decentralized or distributed coord.				Reactive	Distr. MPC[(n)]
Feature / Reference	[16]	[8][(e)]	[17]	[18]	[10][(e)]	[11][(e)]	[12][(e)]	[13]	[9][(e)]	[14]	[15]
Heterogeneous fleets	✓[(a)]	✓	✓	✓	✓[(b)]	✓[(a)]	✓[(a)]	✗	✓	✓[(b)]	✓[(b)]
Avoid environment or path discretization	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
Generic motion planners	✗[(c)]	✓	✓	✓	✓	✗[(c)]	✗[(c)]	✗[(d)]	✓	-[(k)]	✓
Kinodynamic constraints	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓
Asynchronous goal posting	✓	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓
Avoiding static priorities	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓[(l)]
Computation time (coordination)	n.s.	O(C)[(j)]	O(e^{R})	O(C)[(j)]	O(C)[(j)]	O(R)	O(SR)	O(SR)	O(PRS^2)	O($\mathcal{O}$)	n.s.
Scalability (# robots tested in sim.)	n.s.	12	10	30	50	9	100	10	32	100	6
Safe with delays in trajectory execution	✗	✗	✗[(f)]	✓	✓	✓	✓	✓	✗	-[(k)]	✗[(m)]
Safe with unreliable communication	✗	✗	✗[(f)]	✗	✗	✗[(i)]	✗[(g)]	✗[(g)]	✗[(h)]	✗[(k)]	✗[(m)]
Safe with clock de-synchronization	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗[(k)]	✗
Detailed commun. requirements for safety	✗	✗	✗	✗	✗	✗	✗	✗	✗[(p)]	✗	✗
Provable liveness with ideal commun.	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗

a: Assuming robots fit within allotted spatial resources. **b**: Assuming disc-shaped robots. **c**: Graph search w/o motion primitives. **d**: Graph search w/ motion primitives. **e**: Tested only in simulation. **f**: Only for limited, bounded delays. **g**: No formal relationship between communications and speed, sensing neighbor states, or sensing distance provided; unsuccessful sensing/occlusion of neighbors not handled. **h**: Assumed that robots can detect dropped messages; collisions may occur due to occlusions. **i**: Communication radius required to be greater than a specific parameter of the graph. **j**: Limited to critical section computation. **k**: No path is required, but assumes perfect reciprocal visibility and does not ensure absence of collisions. **l**: Assumes static priorities. **m**: Assumes environment free of static obstacles. **n**: Model Predictive Control. **n.s.**: Not specified. **R**: # robots. **C**: # pairwise critical sections (Section II). **S**: # path segments. **P**: # possible equivalent plans. **$\mathcal{O}$**: # obstacles.

TABLE I: Classification of related work according to key requirements of real-world applications [7].

will be relaxed in Sections III and IV, while preserving safety with a computable probability of violation.

Paths and spatial envelopes. Consider a fleet of n (possibly heterogeneous) robots sharing an environment $\mathcal{W} \subset \mathbb{R}^3$. We use $(\cdot)_i$ to indicate that variable $(\cdot)$ refers to robot i. Let $\mathcal{Q}_i$ be the robot's configuration space, and $R_i(q) \subset \mathbb{R}^3$ its collision space when in configuration $q \in \mathcal{Q}_i$. Consider a set of obstacles $\mathcal{O} \subset \mathcal{W}$, so that $\mathcal{Q}_i^{\text{free}} = \{q \in \mathcal{Q}_i : R_i(q) \cap \mathcal{O} = \emptyset\}$ is the set of feasible (i.e., collision free) configurations. Let $\boldsymbol{p}_i : [0,1] \to \mathcal{Q}_i$ be a path in the configuration space parametrized using the arc length $\sigma \in [0,1]$. Then, *path planning* is the problem of finding a (possible executable) path $\boldsymbol{p}_i(\sigma) \in \mathcal{Q}_i^{\text{free}}$ from one feasible starting configuration q^{start} to a final one $q^{\text{goal}} \in \mathcal{Q}_i^{\text{free}}$, such that $q^{\text{start}} = \boldsymbol{p}_i(0)$ and $q^{\text{goal}} = \boldsymbol{p}_i(1)$, typically subject to a set of kinematic constrains $f_i(q, \dot{q}) \leq 0$ (Fig. 1.a). Furthermore, for each $\boldsymbol{p}_i$, the *spatial envelope* $\mathcal{E}_i$ is defined as a set of constraints such that $\cup_{\sigma \in [0,1]} R_i(\boldsymbol{p}_i(\sigma)) \subseteq \mathcal{E}_i$. If the equality holds (which we assume from now on), a spatial envelope is the sweep of the robot's footprint along its path (Fig. 1.b). Henceforth, let $\mathcal{E}_i^{\{\sigma', \sigma''\}} = \cup_{\sigma \in [\sigma', \sigma'']} R_i(\boldsymbol{p}_i(\sigma))$.

Note that $\mathcal{E}_i \cap \mathcal{O} = \emptyset \; \forall i \in \{1, \ldots, n\}$ by construction, that is, collisions between robots and the set of obstacles $\mathcal{O}$ are avoided via path planning. Also, we assume that robots are provided with a low-level safety system for detecting and avoiding obstacles that are not other robots and are not included in $\mathcal{O}$. The focus of the fleet controller proposed in this paper is therefore to avoid inter-robot collisions, not other unforeseen obstacles.

Critical sections. Given a pair of paths $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, collisions may happen only in the set $\{q_i \in \mathcal{Q}_i, q_j \in \mathcal{Q}_j \mid R_i(q_i) \cap \mathcal{E}_j \neq \emptyset \vee R_j(q_j) \cap \mathcal{E}_i \neq \emptyset\}$. In particular, let $\mathcal{C}_{ij}$ be the decomposition of this set into its largest contiguous subsets, each of which is called a *critical section* (Fig. 1.c and 1.d). For each critical section $C \in \mathcal{C}_{ij}$, let $\ell_i^C \in [0,1]$ be the highest value of σ_i before robot i enters C; similarly, let $u_i^C \in [0,1]$ be the lowest value of σ_i after robot i exits C. Considering two temporal profiles $\sigma_i(t)$ and $\sigma_j(t)$, if there exists a time t' such that $R_i(\boldsymbol{p}_i(\sigma_i(t'))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t'))) \neq \emptyset$ (i.e., the robots collide while laying in their envelopes), then $\ell_i^C < \sigma_i(t') < u_i^C$ and $\ell_j^C < \sigma_j(t') < u_j^C$. Hence, given a set of paths $\mathcal{P}$, the *coordination problem* is the problem of synthesizing, for each pair $(i, j \neq i)$ such that $\mathcal{E}_i \cap \mathcal{E}_j \neq \emptyset$, a constraint on temporal profiles $\sigma_i(t)$ and $\sigma_j(t)$ such that $R_i(\boldsymbol{p}_i(\sigma_i(t'))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t'))) = \emptyset$ for all t'. We assume that, when idle, a robot i is placed in a parking position defined by a path $\boldsymbol{p}_i$ of length one. This entails that idle robots are considered in the computation of critical sections.

Precedence constraints and critical points. Precedence constraints are relations among the temporal profiles of two robots. A precedence constraint is a pair $\langle m_i, m_j \rangle$, with $m_i, m_j \in [0,1]$, stating that robot i is not allowed to navigate beyond arc length m_i along its path until robot j has reached arc length m_j along its path — formally, $\sigma_j(t) < m_j \Rightarrow \sigma_i(t) < m_i$. As explained in [18], m_i changes over time to reflect updated precedences and to allow for robots to "follow each other" through critical sections. In general, collisions are avoided if, for each $C \in \mathcal{C}_{ij}$ and for each t, $\sigma_i(t)$ and $\sigma_j(t)$ adhere to the constraint $\langle m_i(t), u_j^C \rangle$, that is, robot i yields for robot j at an appropriately computed arc length $m_i(t)$ along its reference path; this arc length depends on whether robot j has exited critical section C (that is, reached arc length u_j^C) and on its current progress through the critical section:

$$m_i(t) = \begin{cases} \max\left\{\ell_i^C, r_{ij}(t)\right\} & \text{if } \sigma_j(t) \leq u_j^C \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

where $r_{ij}(t)$ is defined as

$$\sup_\sigma \left\{ \sigma \in [\sigma_i(t), u_i^C] : \mathcal{E}_i^{\{\sigma_i(t), \sigma\}} \cap \mathcal{E}_j^{\{\sigma_j(t), u_j^C\}} = \emptyset \right\}. \tag{2}$$

Let $\mathcal{T}$ be the set of precedence constraints regulating the motion of the robots in the fleet. A constraint $\langle m_i, u_j^C \rangle \in \mathcal{T}$ defines unambiguously which robot should yield, where it should yield, and until when yielding is necessary for critical section C. We use $(i <_C j) \in \mathcal{T}$ to indicate that robot j

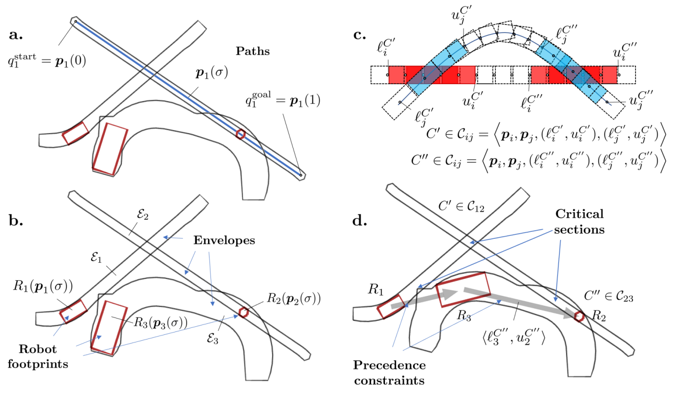
a.
$q_1^{\text{start}} = \boldsymbol{p}_1(0)$
Paths
$\boldsymbol{p}_1(\sigma)$
$q_1^{\text{goal}} = \boldsymbol{p}_1(1)$
b.
$\mathcal{E}_2$
$\mathcal{E}_1$
Envelopes
$R_1(\boldsymbol{p}_1(\sigma))$
$R_2(\boldsymbol{p}_2(\sigma))$
$R_3(\boldsymbol{p}_3(\sigma))$
$\mathcal{E}_3$
Robot footprints
c.
$u_j^{C'}$
$\ell_j^{C''}$
$\ell_i^{C'}$
$u_i^{C''}$
$u_i^{C'}$
$\ell_i^{C''}$
$u_j^{C''}$
$\ell_j^{C'}$
$C' \in \mathcal{C}_{ij} = \left\langle \boldsymbol{p}_i, \boldsymbol{p}_j, (\ell_i^{C'}, u_i^{C'}), (\ell_j^{C'}, u_j^{C'}) \right\rangle$
$C'' \in \mathcal{C}_{ij} = \left\langle \boldsymbol{p}_i, \boldsymbol{p}_j, (\ell_i^{C''}, u_i^{C''}), (\ell_j^{C''}, u_j^{C''}) \right\rangle$
d.
$C' \in \mathcal{C}_{12}$
Critical sections
R_1
$C'' \in \mathcal{C}_{23}$
R_3
R_2
$\langle \ell_3^{C''}, u_2^{C''} \rangle$
Precedence constraints

Fig. 1: Preliminary concepts.

has precedence over robot i at a critical section C. A key feature of the approach is that $\mathcal{T}$ can be updated while robots are in motion. In particular, any heuristic function can be used to determine the precedence constraints in $\mathcal{T}$, as long as a conservative model of each robot's dynamics is employed to filter out ordering decisions that may not be physically realizable (as detailed in [18]).

Let $T_i(t) = \{m_i \mid \exists j : \langle m_i, u_j^C \rangle \in \mathcal{T}(t)\}$ be the set of all the arc lengths at which robot i may be required to yield. We define the *critical point* $\bar{\sigma}_i(t)$ of robot i at time t as the value of σ corresponding to the last reachable configuration along $\boldsymbol{p}_i$ which adheres to the set of constraints $\mathcal{T}(t)$, i.e.,

$$\bar{\sigma}_i(t) = \begin{cases} \arg\min\limits_{m_i \in T_i(t)} m_i & \text{if } T_i(t) \neq \emptyset, \\ 1 & \text{otherwise.} \end{cases} \tag{3}$$

Then, *coordination* is the problem of computing and updating periodically the set of critical points $\bar{\Sigma} = \{\bar{\sigma}_1, \ldots, \bar{\sigma}_n\}$, such that collisions do not occur. Algorithm 1 shows the main body of the supervisory control loop proposed in [18].

Algorithm 1: Coordination at time t.

```
1 sample states[2];
2 if new goals have been posted then
3     update the set of paths $\mathcal{P}$ (using appropriate planners);
4     update the set $\mathcal{C}$ of critical sections;
5 revise the set $\mathcal{T}(t)$ of precedence constraints;
6 compute the set of critical points $\bar{\Sigma}(t)$;
7 communicate changed critical points;
8 sleep until control period $T_c$ has elapsed;
```

Under the assumption of a perfect communication (messages are not delayed or lost), and conservative models of the robots' dynamics, the algorithm ensures that collisions never happen (see [18] for a formal proof). If the assumptions on the channel are removed, then safety no longer holds.

Consider, for instance, a change in the order of access to a critical section $C \in \mathcal{C}_{ij}$ between two consecutive cycles of coordination. Let T_c be the control period of the coordinator, and assume that robot j has precedence over robot i at C at time $t - T_c$. Hence, $\bar{\sigma}_i(t - T_c) = \ell_i^C$ and $\bar{\sigma}_j(t - T_c) > \ell_j^C$. If robot j has not already entered C, i.e., $\sigma_j(t - T_c) < \ell_j^C$, and j can stop before ℓ_j^C according to its dynamic model, then the coordinator may decide to reverse precedence, so that $\bar{\sigma}_j(t) = \ell_j^C$. In this case, if the message to j containing the new critical point $\bar{\sigma}_j(t)$ is delayed or lost, then a collision may happen. The same issue may occur whenever i is starting a new path while j is already driving and $(j <_C i) \in \mathcal{T}$ for some $C \in \mathcal{C}_{ij}$. This undesired situation can occur since the coordination algorithm does not explicitly reason about the non-idealities of the network. The following sections will show how prior knowledge about the channel can be included in the algorithm so that safety is preserved.

[2]Assuming an ideal communication network, current robot states are available via message passing without delays, message loss or disorder.

III. Problem formulation

Given a set of paths $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}$, we aim to define an algorithm to coordinate the fleet via message-passing. The boundary conditions of the multi-robot system are summarized as follows. We assume a duplex point-to-point communication via wireless network (subject to delays and/or message loss) between the coordinator and each agent. Message order can be reconstructed [19] via time-stamps, and message replicas can be filtered out (whether these are sent to increase probability of message reception, or due to multi-path phenomena). Each robot i has a control period of T_i seconds (robots may have different control periods) and sends to the coordinator an update on its state $s_i(t_i)$ sampled within its control period of (clock-driven system). The state report contains the tuple $(q_i(t_i), \dot{q}_i(t_i), \ddot{q}_i(t_i))$, as well as the last critical point $\bar{\sigma}_i$ received by the robot. We also assume that the coordinator receives at least one update of each agent's state every T_c seconds, and that $T_c \geq \max_{i \in 1,\ldots,n} T_i$. Note that robots are not required to be synchronized on a common Coordinated Universal Time (UTC).

A. Channel model

Wireless networks are susceptible to a number of factors that may corrupt packets in transit such as radio frequency interference (RFI), radio signals that are too weak due to distance or multi-path fading, faulty networking hardware, faulty network drivers, or network congestion. Different levels of network-induced imperfections may affect the communication such as time delays, packet loss and disorder, or clock de-synchronization [20]. We model the network considering three parameters: the *maximum bandwidth*, the *packet loss probability*, and the *maximum transmission delay*.

Let $\mathcal{B}(t)$ be the bandwidth of the channel at time t, and let $\mathcal{B}_M$ be the maximum bandwidth (bit/s). Packet loss occurs when one or more packets of data traveling across a network fail to reach their destination. It is measured as the percentage of packets lost with respect to packets sent. We model this phenomenon using a Bernoulli distribution, and assuming the upper-bound of the packet loss probability η to be independent from the identities and locations of the source and destination [20]. Hence, assuming $\mathcal{B}_M = \infty$ (no congestion), the minimal number of replicas (messages containing the same information) required for a successful delivery with probability almost $\bar{p}$ can be computed as

$$N \geq \left\lceil \frac{\log(1-\bar{p})}{\log \eta} \right\rceil. \tag{4}$$

Let $\tau^{ch}_{\max}$ represent the maximum transmission delay (sec), that is, the upper-bound of time elapsed between a send event and the related receive event. This delay is usually the sum of network access delay (i.e., the time required by a queued network packet to be sent out) and the transmission delay through the network medium. While the first is related to the channel's bandwidth, the second is intrinsic of the transmission, and the longer the distance, the longer the delay under the same conditions (bandwidth, protocols, etc.). Also, let $\tau^{ch}_i(t) \leq \tau^{ch}$ be its current realization in a point-to-point communication from/to robot i (Fig. 2). As in [21], we assume $\tau^{ch}_{\max}$ to be constant, symmetric and independent from the source and the destination. Due to congestion, the packet loss probability and the transmission delay may not be independent from the load of the channel. However, for our field of application, it is reasonable to assume the network to be dedicated for fleet management[3]. Hence, given the number of robots, the required bandwidth can be explicitly computed to design the network to avoid congestion (see Section VI). Under this assumption, we assume η and $\tau^{ch}_{\max}$ to be uncorrelated with $\mathcal{B}(t)$.

B. The coordination problem

At each control cycle T_c, the coordinator should decide a *correct* (i.e., collision free) and *feasible* (i.e., physically executable) set of critical points $\bar{\Sigma}$, according to its current view. In doing so, to preserve safety, it should reason about delays affecting the control system. For each robot, we define the maximum delay between sensing and actuation $\tau^{sa}_i = \tau^{sc}_i + \tau^{ca'}_i + \tau^{a'a}_i + \tau^{a}_i$ whose components are defined as follows (see also Fig. 2).

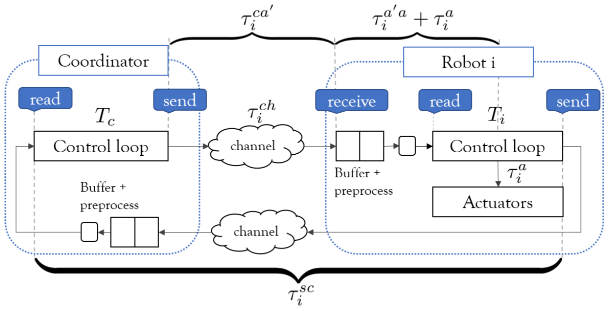

Fig. 2: Delays of a point-to-point communication between the coordinator and each robot i.

τ^{sc}_i is the time elapsed between robot i's state being transmitted and read by the coordinator (sensing delay); assuming robots to be asynchronous, and at least one updated set $\{s_1, \ldots, s_n\}$ to be available within each T_c, then $\tau^{sc}_i = T_c + \tau^{ch} + T_i$ in the worst case. $\tau^{ca'}_i$ is the time required for a critical point $\bar{\sigma}_i$ to be received by robot i; assuming the communication protocol requires to send a burst of M packets for each $\bar{\sigma}_i$, and letting T_p be the period between two consecutive packet deliveries, then $\tau^{ca'}_i = \tau^{ch}_i + (M-1)T_p$; since reasonably $(M-1)T_p \ll T_c$, we assume $\tau^{sc}_i + \tau^{ca'}_i = T_c + 2\tau^{ch} + T_i$ in the worst case. $\tau^{a'a}_i + \tau^{a}_i$ is the delay between when robot i receives a message and the corresponding action is executed (e.g., to yield); this is a function of T_i, the message queue length Q_i, and the robot control system; if $Q_i = 1$ as reasonable, then $\tau^{a'a}_i = T_i$ in the worst case. We also assume no actuation delay, $\tau^a_i = 0$.

[3]so it is not necessary to model the network queue as a Markov process.

Let τ^{break}_i be the maximum breaking time for robot i, which depends on its dynamics and maximum speed. For each $C \in \mathcal{C}_{ij}(t)$, we can assess the feasibility of changing the priority of access to C by looking into the future. In the worst case, a command to yield sent by the coordinator at time t will make the robot stop at time $t + \Delta^{\text{stop}}_i$, where $\Delta^{\text{stop}}_i = \tau^{sa}_i + \tau^{\text{break}}_i$. Hence, if $(i <_C j) \in \mathcal{T}(t - T_c)$, and the coordinator has received the state of robots i and j at times t_i and t_j, it can decide to change the precedence of the two robots to $(j <_C i) \in \mathcal{T}(t)$ iff $\sigma_j(t_j + \Delta^{\text{stop}}_j) \leq \ell^C_j$ (i.e. the robot which loses the priority can effectively stop before entering the critical section). Algorithm 2 implements this feasibility check, given a conservative dynamic model g_i and maximum acceleration/deceleration $u^{\text{acc}}_i, u^{\text{dec}}_i$ of the robot. Specifically,

Algorithm 2: The canStop function.

```
Input: $(q_i(t_i), \dot{q}_i(t_i), \ddot{q}_i(t_i))$ last known state; $\bar{\sigma}_i(t - T_c)$ last
       communicated critical point (or $-1$ if none was communicated);
       $g_i(q_i, \dot{q}_i, \ddot{q}_i, u_i, t)$ dynamic model of the robot; $u_i^{\text{acc/dec}}$
       maximal acceleration/deceleration; $\Delta_i^{\text{stop}}$ look-ahead; $\ell_i^C$
       stopping point at critical section $C$; $\boldsymbol{p}_i$ current path; $\Delta t$
       integration time step.
Output: true iff robot $i$ can stop before entering $C$
1  if $\bar{\sigma}_i(t - T_c) \neq -1 \wedge \bar{\sigma}_i(t - T_c) \leq \ell_i^C$ then return true ;
2  $t \leftarrow t_i$;
3  while $t < t_i + \Delta_i^{\text{stop}}$ do
4      $t' \leftarrow t + \Delta t$;
5      $(q_i(t'), \dot{q}_i(t'), \ddot{q}_i(t')) \leftarrow g_i(q_i(t), \dot{q}_i(t), \ddot{q}_i(t), u_i^{\text{acc}}, t')$;
6      $t \leftarrow t'$;
7  while $\dot{q}_i(t) > 0$ do
8      $t' \leftarrow t + \Delta t$;
9      $(q_i(t'), \dot{q}_i(t'), \ddot{q}_i(t')) \leftarrow g_i(q_i(t), \dot{q}_i(t), \ddot{q}_i(t), u_i^{\text{dec}}, t')$;
10     $t \leftarrow t'$;
11 return $\boldsymbol{p}_i^{-1}(q_i(t)) \leq \ell_i^C$;
```

the robot is clearly capable of stopping (line 1) if it was already constrained to stop at a critical point preceding the beginning of the critical section C at the previous control period (i.e., at time $t - T_c$). If this is not the case, the robot's dynamic model g_i is used to compute whether it can achieve zero velocity before the critical point (lines 3–10). In doing so, it assumes that the robot has progressed with maximal acceleration from its last reported state $(q_i(t_i), \dot{q}_i(t_i), \ddot{q}_i(t_i))$ for a period of Δ^{stop}_i (lines 3–6). Note that a robot state report with $\dot{q}(t_i) = 0$ is not sufficient to conclude that the robot can stop moving, as this state was sampled at time t_i and the robot may have started moving in the meantime.

IV. The coordination algorithm

In (1) and (2), we assume that r_{ij} (the furthest σ_i the yielding robot is allowed to reach) is computed with knowledge of the current progress $\sigma_i(t)$, $\sigma_j(t)$ of the two robots. Due to possible delays in communication, we must now rely on $\sigma_i(t_i)$ and $\sigma_j(t_j)$, where $t - \tau_i^{sc} \le t_i < t$ and $t - \tau_j^{sc} \le t_j < t$. Hence,

$$m_i(t) = \begin{cases} \max\left\{\ell_i^C, r_{ij}(t)\right\} & \text{if } \sigma_j(t_j) \le u_j^C \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

and $r_{ij}(t)$ is computed as

$$\sup_\sigma \left\{\sigma \in [\sigma_i(t_i), u_i^C] : \mathcal{E}_i^{\{\sigma_i(t_i),\sigma\}} \cap \mathcal{E}_j^{\{\sigma_j(t_j),u_j^C\}} = \emptyset\right\}. \tag{6}$$

With this revised formulation of precedence constraints, we can now define an algorithm for coordination which guarantees correct behavior of the fleet in the presence of non-ideal communication channels. Algorithm 3 is based on the principle of Algorithm 1: at each T_c, the states s_i of all robots are updated with the last report received, each at a potentially different time t_i (line 4); paths are computed for idle robots for which a new goal has been posted (lines 5–9); critical sections are updated (lines 10–11); and constraints are revised (lines 12–13) and communicated to the robots (line 14). At the core of the coordination algorithm is a call to the function *revise*, detailed in Algorithm 4. For each critical section, this function decides the order of traversal, according to the current state of the involved robots (lines 2–8), using the aforementioned *canStop* function (see Algorithm 2). Then, it updates the set of precedence constraints $\mathcal{T}(t)$ according to (5) and (6) (lines 9–10).

Algorithm 3: The coordination algorithm.

```
Input: $\mathcal{G}$ set of goals posted for robots $\{1,\dots,n\}$.
1  $\mathcal{P} \leftarrow \emptyset, \mathcal{C} \leftarrow \emptyset, \mathcal{T} \leftarrow \emptyset, \bar{\Sigma} \leftarrow \{-1\}^n$;
2  while true do
3      $t \leftarrow$ getCurrentTime();
4      for $i \in [1,\dots,n]$ do $s_i(t_i) \leftarrow$ getStatusMsg($i$) ;
5      for $i : g_i \in \mathcal{G} \wedge$ isIdle($s_i(t_i)$) do
6          $\mathcal{G} \leftarrow \mathcal{G} \setminus \{g_i\}$;
7          remove robot $i$ from $\mathcal{P}$, $\bar{\Sigma}(t - T_c)$ and $\mathcal{C}$;
8          $\boldsymbol{p}_i \leftarrow$ computePath($s_i(t_i), g_i$);
9          $\mathcal{P} \leftarrow \mathcal{P} \cup \{\boldsymbol{p}_i\}$;
10         for $(\boldsymbol{p}_i, \boldsymbol{p}_{j\neq i}) \in \mathcal{P}^2$ do
11             $\mathcal{C} \leftarrow \mathcal{C} \cup$ getIntersections($\mathcal{E}_i(\boldsymbol{p}_i), \mathcal{E}_j(\boldsymbol{p}_j)$);
12     $\mathcal{T}(t) \leftarrow$ revise($\mathcal{P}, \mathcal{C}, \mathcal{T}(t-T_c), \bar{\Sigma}(t-T_c), \{s_1(t_1),\dots,s_n(t_n)\}$);
13     $\bar{\Sigma}(t) \leftarrow \forall i$ compute $\bar{\sigma}_i$ as in (3) or -1 if $i$ is idle;
14     for $i \in [1,\dots,n]$ do send($i, \bar{\sigma}_i(t)$) ;
15     $\Delta t =$ getCurrentTime() $- t$;
16     if $\Delta t < T_c$ then sleep($\Delta t$) ;
```

In particular, for each critical section C, if neither of the two involved robots is known to have passed the critical section's upper bound (line 3), and both of them can achieve zero velocity before entering it (line 4), then an ordering is heuristically[4] decided. If one of the two robots has entered or cannot stop before entering C (lines 6–7), then that robot is given precedence. If this is the case for both robots, the previously decided ordering is re-imposed (line 8). Note that an idle robot involved in a critical section is necessarily already inside it, hence, will always have precedence over the other robot.

[4]Note that, as in [18], any heuristic can be chosen here.

Algorithm 4: The revise function.

```
Input: $\mathcal{P}$ current set of paths; $\mathcal{C}$ (possibly empty) set of pairwise
       critical sections; $\mathcal{T}(t-T_c)$ (possibly empty) previous set of
       precedence constraints; $\bar{\Sigma}(t-T_c)$ previous set of critical
       points; $s_i(t_i)$ last received robot state, including $\sigma_i(t_i)$.
Output: $\mathcal{T}_{\text{rev}}$ set of revised precedence constraints.
1  $\mathcal{T}_{\text{rev}} \leftarrow \emptyset$;
2  for $\mathcal{C}_{ij} \in \mathcal{C}, C \in \mathcal{C}_{ij}$ do
3      if $\sigma_i(t_i) < u_i^C \wedge \sigma_j(t_j) < u_j^C$ then
4          if $\sigma_i(t_i + \Delta_i^{\text{stop}}) \le \ell_i^C \wedge \sigma_j(t_j + \Delta_j^{\text{stop}}) \le \ell_j^C$ then
5              $(h <_C k) \leftarrow$ compute ordering with a heuristic;
6          else if $\sigma_i(t_i + \Delta_i^{\text{stop}}) > \ell_i^C \wedge \sigma_j(t_j + \Delta_j^{\text{stop}}) \le \ell_j^C$ then
               $(h <_C k) \leftarrow (j <_C i)$ ;
7          else if $\sigma_i(t_i + \Delta_i^{\text{stop}}) \le \ell_i^C \wedge \sigma_j(t_j + \Delta_j^{\text{stop}}) > \ell_j^C$ then
               $(h <_C k) \leftarrow (i <_C j)$ ;
8          else $(h <_C k) \leftarrow$ get previous ordering from $\mathcal{T}$ ;
9      $\langle m_h, u_k^C\rangle \leftarrow$ compute as in (5) and (6);
10     $\mathcal{T}_{\text{rev}} \leftarrow \mathcal{T}_{\text{rev}} \cup \{\langle m_h, u_k^C\rangle\}$;
11 return $\mathcal{T}_{\text{rev}}$;
```

V. Safety Analysis

In this analysis, we make the following assumptions:

A1. Paths do not start or end in critical sections.
A2. Robots always stay within their envelopes.
A3. The channel delay τ_i^{ch} is bounded.

A1 and A2 are made to simplify the analysis — the algorithm can be easily modified to include these specific cases. A3 is a reasonable assumption to make on any real network. We start by considering $\eta = 0$ (no packet loss), while the case of $\eta > 0$ is considered in Section VI.

Lemma 1. *Algorithms 4 and 2 satisfy the preposition:* if $\mathcal{T}(t')$ *is a set of feasible constraints,* then $\mathcal{T}(t'')$ *is a set of feasible constraints* $\forall t'' > t'$.

Proof. The proof is given by induction. $\mathcal{T}(0)$ is feasible because robots do not move if $\bar{\Sigma} = \{-1\}^n$ and we assume A1. Thanks to Algorithm 2, precedences can be changed only if the yielding robot i can effectively stop before ℓ_i^C.

Basic step: $\mathcal{T}(0)$ feasible $\Rightarrow$ $\mathcal{T}(T_c)$ feasible. The proof is given by contradiction: note that $\mathcal{T}(T_c)$ is unfeasible iff $\exists\langle m_i, u_j^C\rangle \in \mathcal{T}(T_c)$ such that $m_i(T_c) < \bar{\sigma}_i(0)$ and, given $t_j \in [0, T_c)$, i cannot stop at

$$m_i(T_c) = \begin{cases} \max(\ell_i^C, r_{ij}(T_c)) \text{ if } \sigma_j(t_j) \le u_j^C & \text{(c1)} \\ 1 \qquad\qquad\qquad \text{otherwise} & \text{(c2)} \end{cases}$$

Condition (c1) can happen only in the following cases:

a. $(i <_C j) \in \mathcal{T}(0) \wedge (i <_C j) \in \mathcal{T}(T_c)$, i.e., the previously-decided order is held.
b. $(j <_C i) \in \mathcal{T}(0) \wedge (i <_C j) \in \mathcal{T}(T_c)$, i.e., the previously-decided order is changed.
c. $C \notin \mathcal{C}(0) \wedge C \in \mathcal{C}(T_c)$, i.e., the order is decided for the first time.

However, feasibility holds for all of them: a. Since σ_j is monotone increasing, then according to (6) $\max(\ell_i^C, r_{ij}(0)) \le \max(\ell_i^C, r_{ij}(T_c))$. But then, $m_i(T_c)$ is feasible since $\bar{\sigma}_i(0) \le m_i(0) \le m_i(T_c)$. b. $\langle m_j, u_i^C\rangle \in \mathcal{T}(0)$, and according to A1, $m_j(0) = \ell_j^C$. Moreover, due to (3), $\bar{\sigma}_j(0) \le \ell_j^C$, so at time T_c robot j can stop. Hence, according to Algorithm 4, $(i <_C j) \in \mathcal{T}(T_c)$ iff $\sigma_i(T_c + \Delta_i^{\text{stop}}) \le \ell_i^C$, i.e., iff the constraint is feasible. c. Robot j is assigned to a new goal at time T_c, so it is assumed to be not in motion (idle). Then, according to A1, $(j <_C i) \in \mathcal{T}(T_c)$ is feasible, so as in the previous case, $(i <_C j) \in \mathcal{T}(T_c)$ may be decided iff feasibility holds.

In condition (c2), by definition, $\bar{\sigma}_i(t) \in [0, 1]$, and since i can stop at $\bar{\sigma}_i(0) \leq 1$, then $\bar{\sigma}_i(T_c) = 1$ is feasible. Hence, $\nexists \langle m_i, u_j^C \rangle \in \mathcal{T}(T_c)$ that is unfeasible if $\mathcal{T}(0)$ is feasible, proving the preposition in the basic step.

Inductive step: Note that the previous proof holds for every consecutive pair of $\mathcal{T}(t')$ and $\mathcal{T}(t' + T_c)$, if $\mathcal{T}(t')$ is feasible. Then, $\mathcal{T}(0)$ feasible $\Rightarrow \mathcal{T}(T_c)$ feasible $\Rightarrow \cdots \Rightarrow \mathcal{T}(kT_c)$ feasible, $\forall k \in \mathbb{N}^+$. □

As a result, we can prove that:

Theorem 1 (Feasibility)**.** *The set* $\bar{\Sigma}(t)$ *is feasible.*

Proof. According to (3), $\mathcal{T}(t)$ feasible implies $\bar{\Sigma}(t)$ feasible. Hence, the proof follows from Lemma 1. □

We use this result to prove the correctness of Algorithm 3:

Theorem 2 (Correctness)**.** *The set* $\bar{\Sigma}(t)$ *is collision free.*

Proof. A collision happens iff $\exists t$ such that $R_i(\boldsymbol{p}_i(\sigma_i(t))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t))) \neq \emptyset$. According to A2, this may happen iff both robots are inside a critical section. Hence, correctness holds if, for every $C \in \mathcal{C}_{ij}(t)$, Algorithm 4 ensures collision-free access to C (c1), and (5) and (6) ensure collision-free progress through C (c2).

Case c1. If no goals are posted at time 0, then robots are all idle and there are no critical sections due to A1. This is a safe starting configuration. Let t_0 be the time such that $g_i \in \mathcal{G}(t_0)$ is assigned to robot i. Let the set of robots for which $\exists C \in \mathcal{C}_{ij}(t_0)$ be partitioned in the sets $\mathcal{D}$ containing the robots which are already driving, and $\mathcal{I}$ containing the robots which are still idle. We can prove that $R_i(\boldsymbol{p}_i(\sigma_i(t_0))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t_0))) = \emptyset$, $\forall j \in \mathcal{I}$ as follows. According to Algorithm 2, $\sigma_i(t_0 + \Delta_i^{\text{stop}}) \leq \ell_i^C$ (robot i can stop) and robot j is idle, so, Algorithm 4 will decide for $\langle \ell_i^C, u_j^C \rangle$ according to (5) and (6). Note that, since i is still not moving, feasibility holds at time t_0. Moreover, (3) will ensure $\bar{\sigma}_i(t) \leq \ell_i^C \ \forall t$ as long as j remains idle. Finally, due to A1, we can assume that j would not be assigned a goal $\forall t \geq t_0$, preventing j to collide with robot i. We can also prove that $R_i(\boldsymbol{p}_i(\sigma_i(t_0))) \cap R_j(\boldsymbol{p}_j(\sigma_j(t_0))) = \emptyset$, $\forall j \in \mathcal{D}$, as follows. According to A1, we can assume $R_i(\boldsymbol{p}_i(0)) \cap \mathcal{E}_j(t_0) = \emptyset$. Note that at time t_0 there exists at least a feasible and correct ordering $(i <_C j) \in \mathcal{T}(t_0)$, $\forall C \in \mathcal{C}_{ij}(t_0)$. Also, Algorithm 4 may decide for $(j <_C i) \in \mathcal{T}(t_0)$ iff $\sigma_j(t_0 + \Delta_j^{\text{stop}}) \leq \ell_j^C$. Hence correctness holds: due to feasibility, mutual access to the critical section is collision-free, as either $\bar{\sigma}_i(t_0) \leq \ell_i^C \Rightarrow \sigma_i(t_0 + T_c) \leq \ell_i^C$, or $\bar{\sigma}_j(t_0) \leq \ell_j^C \Rightarrow \sigma_j(t_0 + T_c) \leq \ell_j^C$. Also, at time $t > t_0$ the previously decided ordering can be changed iff the yielding robot can stop, so correctness holds even for $t > t_0$.

Case c2. Let $(i <_C j) \in \mathcal{T}(t)$ be the order of accessing a critical section $C \in \mathcal{C}_{ij}(t)$. Assume that the last received status messages reports that the robots are both inside C. It is easy to show that safety is preserved if the status message of either robot is delayed. Let $\sigma_i(t_i)$ and $\sigma_i(t)$ be the last notified and the current value of σ of robot i respectively (the same for j). Since σ is a monotone increasing function, then for any interval $[\sigma_i', \sigma_i'']$, $\sigma_i'' > \sigma_i'$, if $\mathcal{E}_i^{\{\sigma_i', \sigma_i''\}} \cap \mathcal{E}_j^{\{\sigma_j(t_j), u_j^C\}} = \emptyset$, then $\mathcal{E}_i^{\{\sigma_i', \sigma_i''\}} \cap \mathcal{E}_j^{\{\sigma_j(t), u_j^C\}} = \emptyset$, that is, a delay of the leading robot preserves safety. Moreover, given $\sigma_j(t_j)$, (6) will give the same $r_{ij}(t)$, $\forall \sigma_i : \ell_i^C < \sigma_i < u_i^C$, that is, the same conclusion holds in case of a delay of the waiting robot. □

The proposed algorithm thus maintains a key feature of [18] even in the case of arbitrary (bounded) channel delays:

Corollary 1. *For any robot i in the fleet, any realization of $\sigma_i(t)$ that adheres to $\bar{\Sigma}(t)$ is correct. This includes unforeseen stops or changes in velocity due to low-level control and/or safety mechanisms.*

VI. The communication protocol

In order to minimize the possibility of robots colliding due to packet loss ($\eta \geq 0$), we could use a protocol with message acknowledgment. Although this would ensure the deterministic outcome of message sending, a single non-acknowledged message could invalidate the consistency of the set $\bar{\Sigma}(t)$. Thus, we would have to modify the algorithm in a non-trivial (and potentially computationally expensive) manner in order to properly handle such situations. We adopt here an approach which preserves the relative simplicity of the algorithm, namely, an unreliable (UDP-like) protocol. Specifically, given a model of the channel, this can be used to compute the smallest N such that a burst of N equal messages will result in a probability of successful delivery that is greater than a threshold $\bar{p}$.

Assume that $\langle m_i, u_j^C \rangle \in \mathcal{T}(t - T_c)$, $\langle m_j, u_i^C \rangle \in \mathcal{T}(t)$, and that $\bar{\sigma}_j(t) = m_j$. According to (4), in the worst case,

1. a correct change of priority may happen with probability $\bar{p}^2$ (i.e., both $\bar{\sigma}_i(t)$ and $\bar{\sigma}_j(t)$ are successfully delivered);
2. the probability of maintaining the old constraint is equal to $(1 - \bar{p})^2$ (i.e., both $\bar{\sigma}_i(t)$ and $\bar{\sigma}_j(t)$ are lost);
3. a collision may happen with probability $\bar{p}(1 - \bar{p})$ (i.e., $\bar{\sigma}_i(t)$ is successfully delivered while $\bar{\sigma}_j(t)$ is lost);
4. a temporary starvation may happen with probability $\bar{p}(1 - \bar{p})$ (i.e., $\bar{\sigma}_i(t)$ is lost while $\bar{\sigma}_j(t)$ is successfully delivered).

We consider a constraint to be violated whenever a communication fails to happen as it would without packet loss; hence, the probability of constraint violation is $\bar{p}_u = 1 - \bar{p}^2$. Note that constraint violations do not necessarily lead to collisions.

Bandwidth. As mentioned in Section III-A, congestion can be avoided by explicitly considering the maximum bandwidth required for successful coordination. Specifically, at each time t the required bandwidth is given by:

$$\mathcal{B}(t) = \sum_{i=1}^{n_d(t)} N_i \frac{b_i}{T_i} + N_c(t) N \frac{b_c}{T_c},$$

where N_i is the number of replicas sent by robot i every T_i; b_i is the number of bits of each s_i message; b_c is the number of bits of each $\bar{\sigma}_i$ message; $N_c(t)$ is the number of $\bar{\sigma}_i(t)$ that are updated at time t; and $n_d(t)$ is the number of driving robots at time t. Then, the maximum load of the network can be computed assuming $N_c(t) = n_d(t) = n$ (i.e., all robots are driving and all the critical points are updated). However, the subset of $\bar{\Sigma}(t)$ that is effectively communicated may be smaller (since only the changes with respect to $\bar{\Sigma}(t-T_c)$ are really informative), and the effective load may be lower.

Let $\gamma \in [0,1]$ be a desired percentage of bandwidth dedicated to coordination[5]. Assuming (b_i, T_i, N_i) to be equal for each robot, congestion is avoided if

$$n\left(N_i \frac{b_i}{T_i} + N\frac{b_c}{T_c}\right) \leq \gamma \mathcal{B}_M. \tag{7}$$

We can use (7) to relate the number of robots that can be coordinated to the maximum bandwidth $\mathcal{B}_M$, with a probability of constraint violation lower than a the given threshold. The goal is then to define a function to compute the maximum n assuming as parameters the maximum bandwidth $\mathcal{B}_M$, the control periods T_i, T_c, the upper bound of packet loss probability η, and a desired threshold for the probability of constraint violation. For this purpose, we define $\alpha \in \mathbb{R}^+$ such that $T_c = \alpha T_i$; hence, $N_i = \lceil N/\alpha \rceil$ is the minimum number of replicas needed to ensure that at least one s_i will be received from each robot at each T_c with probability almost $\bar{p}$. Then, from (7), we have that:

$$n^{max} = \left\lfloor \frac{\mathcal{B}_M T_i}{b_i + b_c} \right\rfloor, \text{ hence } n(\alpha, \gamma, N) = \gamma \frac{\alpha}{N} n^{max}.$$

Decreasing α, T_c, and Δ_i^{stop} allows to react faster to changes, confirming the intuitive fact that a fleet that reacts faster to changes also imposes a higher average load on the network. α can be used by the designer to tune this trade-off as desired, possibly defining an optimization problem.

Controller Synthesis. The following process can be used to design a fleet controller that accounts for a given channel model $(\mathcal{B}_M, \gamma, \eta, T_i, \alpha)$: (i) choose the desired upper-bound $\bar{p}_u$ on the probability of constraint violation; (ii) compute $\bar{p}$ so that $\bar{p}_u \leq 1 - \bar{p}^2$; (iii) compute the minimal number of replicas needed to ensure a probability of receiving at least one replica greater than $\bar{p}$ as $N \geq \lceil \log(1-\bar{p})/\log \eta \rceil$; (iv) set $T_c = \alpha T_i$ and $N_i = \lceil N/\alpha \rceil$. Then, the maximal number n of robots which can be coordinated using a UDP-like protocol is given by $n = \gamma \frac{T_c}{N} \frac{\mathcal{B}_M}{b_i + b_c}$.

VII. Experimental validation

The implementation of Algorithm 3 evaluated here maintains the original good properties of the one proposed in [18], hence fulfills all requirements stated in Table I. The computational overhead of the approach remains unchanged, as the scalability analyses of [18] remain valid. Hence, the simulations presented here focus on communication, and, in particular, aim to confirm the formal properties stated in the previous theoretical discussion. In all experiments, Algorithm 3 was run on an Intel Core i7-5500U CPU @ 2.40GHz × 4 processor. The algorithm is implemented in Java and is available as open source [22].

Setup. The simulator back-end presented in [18] was used for all tests. Uniformly distributed random variables were used for injecting different realizations of $\tau_i^{ch}(t) \in \left[\tau_{\min}^{ch}, \tau_{\max}^{ch}\right]$. For simplicity, simulations considered homogeneous robots (although Algorithm 3 is designed for general heterogeneous platforms). Paths were computed using a sampling-based motion planner (RRTConnect). Robot motion synthesis and the conservative model g_i used in Algorithm 2 were both based on a trapezoidal velocity profile with maximum velocity $v_i^{\max}$ and constant acceleration/deceleration $u_i^{\text{acc}} = u_i^{\text{dec}} = u_i^{\max}$. Goals were dispatched asynchronously to robots, requiring them to navigate 10 times from their current location to the opposite side of the environment and back. Deadlocks were handled by re-planning, via a prioritized planning method [23]. Simulation parameters are listed in Table II, and tests were repeated 10 times to obtain statistically significant results. Moments of all tests are shown in the video attachment.

[5]Usually, $\gamma < 1$ to reserve bandwidth for synchronization or QoS.

	Test 1	Test 2
Environment	empty space	Map 1 and 2
Motion planning	off-line	online
Channel model:		
η	0	0.2
$\tau_{\max}^{\text{ch}}$	[0.1, 2] sec	2 sec
$\tau_{\min}^{\text{ch}}$	$\tau_{\max}^{\text{ch}}$	0.01 sec

In all tests: $T_c = 1\text{s}, T_i = 0.03\text{s}, v_i^{\max} = \pm 4\text{m/s}, u_i^{\max} = \pm 3\text{m/s}^2$.

TABLE II: Simulation parameters.

Evaluation metrics. Given the probability $\bar{p}_u$ (and so, the maximum $\bar{p} = \sqrt{1-\bar{p}_u}$), the probability of the system being in an unsafe state is upper bounded by $\bar{p}(1-\bar{p})$ (i.e., the message to the newly yielding robot is lost, not the other). This situation involves a pair of events, and is difficult to measure in a distributed setting. Conversely, if a collision happens, then it is certain that the previous pair of events has indeed occurred. Hence, defining as collision rate the ratio between the number of collisions observed and the number of critical sections traversed, we expect the collision rate to be less than $\bar{p}(1-\bar{p})$. It should be remarked that the formal proof of this is given in the previous sections; this experimental evaluation is intended to support the theoretical findings. Note also that the collision rate tends to $\bar{p}(1-\bar{p})$ as the number of observations approaches infinity.

A. Test 1: Injecting channel delays ($\tau_i^{ch} > 0$, $\eta = 0$).

The goal is to validate the claim that prior knowledge of the channel delay is required to ensure safety. The collision rate obtained using the algorithm proposed in [18] (which is *uninformed* of the channel model) is compared with the one obtained using our implementation of Algorithm 3. To provide the same testing conditions, the simulation uses a fixed set of paths and a constant channel delay ($\tau_i^{ch} = \tau_{\max}^{ch}$, $\forall i$). Results are shown in Fig. 3, which highlight the unsafety of the uninformed algorithm [18] and validate the safety of the proposed solution.

B. Test 2: Random Paths ($\tau_i^{ch} > 0$, $\eta > 0$)

The goal is to validate safety when the network's non-idealities are modeled as described in Section III-A. In order to stress Algorithm 3 as much as possible, we provoke random delays and packet loss rate within an upper bound, and generate random paths for the robots (so that the geometry of critical sections is unpredictable). We simulate a particularly

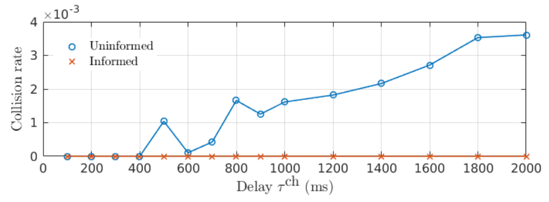

Fig. 3: Test 1: collision rate using the uninformed algorithm [18] and the one proposed in this article.

bad communication channel, with high packet loss and high delays with great variance (which causes the "jerky" motions visible in the video). Simulations are run considering two environments, with a total amount of critical sections analyzed equal to 110693 (Map 1) and 42748 (Map 2). Results are shown in Table III; as expected, the measured collision, packet and message loss rates are smaller then the expected values.

$\bar{p}_u = 2\%$	Upper bound	Measured		
		avg.	max	min
Packet Loss	0.2	0.201	0.209	0.191
Message Loss	1.0e-2	8.9e-3	1.3e-2	4.6e-3
Collision Rate	9.9e-3	3.8e-4	1.3e-3	0
$\bar{p}_u = 10\%$	Upper bound	Measured		
		avg.	max	min
Packet Loss	0.2	0.182	0.192	0.176
Message Loss	5.1e-2	3.6e-2	4.2e-2	3.1e-2
Collision Rate	4.9e-2	3.1e-4	8.5e-4	0

$\bar{p}_u = 2\%$	Upper bound	Measured		
		avg.	max	min
Packet Loss	0.2	0.198	0.209	0.190
Message Loss	1.0e-2	8.3e-3	1.1e-2	6.0e-3
Collision Rate	9.9e-3	4.9e-4	1.1e-3	0
$\bar{p}_u = 10\%$	Upper bound	Measured		
		avg.	max	min
Packet Loss	0.2	0.185	0.202	0.149
Message Loss	5.1e-2	3.5e-2	4.2e-2	2.6e-2
Collision Rate	4.9e-2	4.4e-4	1.2e-3	0

TABLE III: Test 2: results using Map 1 (top) and 2 (bottom).

VIII. Conclusion

We have presented a centralized coordination algorithm (based on [18]) which does not assume perfect communication, allowing message disorder, bounded message delays, and message loss. The approach enables periodic, heuristically-guided constraint revision while guaranteeing a desired maximum rate of constraint violation. We have shown formally that this probability is zero with no packet loss and arbitrary delay. We have also provided a set of rules to relate communication infrastructure requirements, number of robots, controller parameters, and maximum rate of constraint violation. These rules can be used for fleet controller synthesis, as well as for fleet and network dimensioning. We have validated our formal findings quantitatively via simulations. Preliminary tests with two real robots subject to random communication delays and packet loss confirm[6] the applicability of the method to real-world use-cases. Future work will focus on deployment in environments that are affected by severe network problems (e.g., quarries and underground mines). We will also analyze the problem of deadlock avoidance and resolution.

[6]Video available at https://youtu.be/-rBK_Qgcj28.

References

[1] V. M. G. Martínez *et al.*, "Ultra reliable communication for robot mobility enabled by sdn splitting of wifi functions," in *IEEE Symp. Comput. Commun.*, 2018.

[2] H. Kunsei, K. S. Bialkowski, M. S. Alam, and A. M. Abbosh, "Improved communications in underground mines using reconfigurable antennas," *IEEE Trans. Antennas Propag.*, vol. 66, no. 12, pp. 7505–7510, 2018.

[3] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, and L. Zhao, "Vehicle-to-everything (V2X) services supported by LTE-based systems and 5g," *Commun. Surveys Tuts.*, vol. 1, no. 2, pp. 70–76, 2017.

[4] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Commun. Surveys & Tutorials*, vol. 20, no. 2, pp. 978–1013, 2018.

[5] V. Milanes, J. Villagra, J. Godoy, J. Simo, J. Pérez, and E. Onieva, "An intelligent V2I-based traffic management system," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 49–58, 2012.

[6] E. Nett and S. Schemmer, "Reliable real-time communication in cooperative mobile applications," *IEEE Trans. Comput.*, vol. 52, no. 2, pp. 166–180, 2003.

[7] H. Andreasson *et al.*, "Autonomous transport vehicles: where we are and what is missing," *IEEE Robot. Autom. Mag.*, vol. 22, no. 1, pp. 64–75, 2015.

[8] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *Int. J. Robot. Research*, vol. 24, no. 4, pp. 295–310, 2005.

[9] K. E. Bekris, D. K. Grady, M. Moll, and L. E. Kavraki, "Safe distributed motion coordination for second-order systems with different planning cycles," *Int. J. Robot. Research*, vol. 31, no. 2, pp. 129–150, 2012.

[10] M. Čáp, J. Gregoire, and E. Frazzoli, "Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Syst.*, 2016, pp. 5113–5118.

[11] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "A decentralized control strategy for the coordination of AGV systems," *Control Eng. Practice*, vol. 70, pp. 86–97, 2018.

[12] S. Manca, A. Fagiolini, and L. Pallottino, "Decentralized coordination system for multiple agvs in a structured environment," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 6005–6010, 2011.

[13] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević, and S. Bogdan, "Decentralized control of multi-AGV systems in autonomous warehousing applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, 2016.

[14] D. Bareiss and J. Van den Berg, "Generalized reciprocal collision avoidance," *Int. J. Robot. Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[15] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Syst.*, 2017, pp. 236–243.

[16] N. Smolic-Rocak, S. Bogdan, Z. Kovacic, and T. Petrovic, "Time windows based dynamic routing in multi-AGV systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 151–155, 2010.

[17] F. Pecora, M. Cirillo, and D. Dimitrov, "On mission-dependent coordination of multiple vehicles under spatial and temporal constraints," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots & Syst.*, 2012, pp. 5262–5269.

[18] F. Pecora, H. Andreasson, M. Mansouri, and V. Petkov, "A loosely-coupled approach for multi-robot coordination, motion planning and control," in *Proc. 28th Int. Conf. Autom. Planning & Scheduling*, 2018.

[19] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, no. 7, pp. 558–565, 1978.

[20] L. Zhang, H. Gao, and O. Kaynak, "Network-induced constraints in networked control systems—a survey," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 403–416, 2013.

[21] J. Nilsson, "Real-time control systems with delays," Ph.D. dissertation, Lund Institute of Technology Lund, Sweden, 1998.

[22] F. Pecora and A. Mannucci, *An online multi-robot coordination algorithm based on trajectory envelopes (branch udp)*, 2019, https://github.com/FedericoPecora/coordination_oru.

[23] M. Čáp, P. Novák, A. Kleiner, and M. Seleckỳ, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, 2015.