COORP: A High Performance and High Accuracy Wireless Supporting System for Coordinated Robotic Learning

Shengliang Deng†, Xiuxian Guan, Zekai Sun, Shixiong Zhao†, Tianxiang Shen, Xusheng Chen, Tianyang Duan, Yuexuan Wang, Jia Pan, Yanjun Wu, Libo Zhang, and Heming Cui*, *Member, IEEE*

Abstract

Coordinated robotic learning is a new paradigm where multiple robots connect with a wireless network, make real-time coordinated decisions using a control model, and utilize the distributed computation resources in the robot group to train the control model to fast adapt to new environments. In this new paradigm, coordinated decisions should be transmitted immediately to meet real-time constraints, while enough bandwidth should be left for distributed model training. Unfortunately, existing wireless network supporting systems, including prioritized contention systems and global planning systems, are unable to meet these two requirements simultaneously. We present COORP to meet both requirements for coordinated robotic learning based on our new abstraction, coordinated preemption. COORP coordinates all robots to minimize simultaneously transmitting bandwidth-hungry flows such that latency-sensitive coordinated decisions immediately acquire the shared wireless channel on demand. Further, COORP constructs a virtual-preemption layer for existing commodity wireless network interface cards to enable latency-sensitive flows to get transmitted before bandwidth-hungry flows on the same robot. Evaluation shows that COORP achieves a low violation rate (8.8%) of reaction time boundary with only a minor slowdown (13%~16%) of the training process compared to EDCA, guaranteeing high performance and high accuracy of coordinated robotic learning.

*Corresponding author.

†IEEE student member.

Email: sldeng@cs.hku.hk, guanxiux@hku.hk, {zksun, sxzhao, txshen2, xschen}@cs.hku.hk, tianyang.duan@ucdconnect.ie, {amywang, jpan}@cs.hku.hk, {yanjun, libo}@iscas.ac.cn, heming@cs.hku.hk

I. INTRODUCTION

Coordinated robotic learning enables a group of robots to closely coordinate with each other and adapt to new environments, and is increasingly important to mission-critical multi-robot applications in field, including rescue [1], navigation [2], and surveillance [3], [4]. Despite each robot's local functionalities (e.g., collision detection), we summarize that a typical coordinated robotic learning workflow [1], [2], [4]–[9] comprises two global coordinations: a *multi-view perception-control coordination* (MPC) that continuously exchanges information among robots and makes coordinated decisions, and a *distributed reinforcement learning coordination* (DRLC) that exploits the distributed computation power of all robots to adapt to new environments.

The MPC of a robot group loops over *perception exchange*, *inference*, and *control dissemination*, as shown in Figure 1. During *perception exchange*, a leader robot collects and combines perceptions (e.g., images) captured by the sensors (e.g., cameras) of all the other robots, via a wireless network for multi-view awareness of the environment. During *inference*, the leader robot feeds the combined perception to a deep neural network control model which takes environmental information as input and produces cooperative control messages (e.g., velocities) as output. After that, the leader robot *disseminates* the control messages to each robot for the next move.

The DRLC is usually spawned when a robot group enters a new environment [5]. As the hardware resources per robot are limited, data parallelism (DP, see Section II-A) is usually adopted to exploit the distributed computation power of all the robots. A data parallel DRLC consists of *data dissemination*, *parallel training*, and *parameter synchronization*. Note that, in each iteration of the MPC loop, the leader robot collects a composed data of a combined perception, control messages that contain actions, and a corresponding reward that reflects the effect of the actions. When a batch of data is accumulated, the leader robot partitions the data batch and *disseminates* each robot a partition. Then each robot trains the control model *in parallel*. The leader robot gathers the parameter updates computed by each robot to *synchronize* the control model. DRLC runs until the robot group adapts to the new environment. In this paper, we focus on coordinated robotic learning with one MPC and one DRLC.

We identify two stringent requirements for mission-critical coordinated robotic learning applications. First, the MPC loop requires *fast reaction* (**R1**): each iteration of MPC loop should finish within a tight reaction time boundary (e.g. 33ms [10]–[12]) to avoid severe flaws like robot collisions demonstrated in Figure 1. Second, the DRLC requires *fast model convergence*

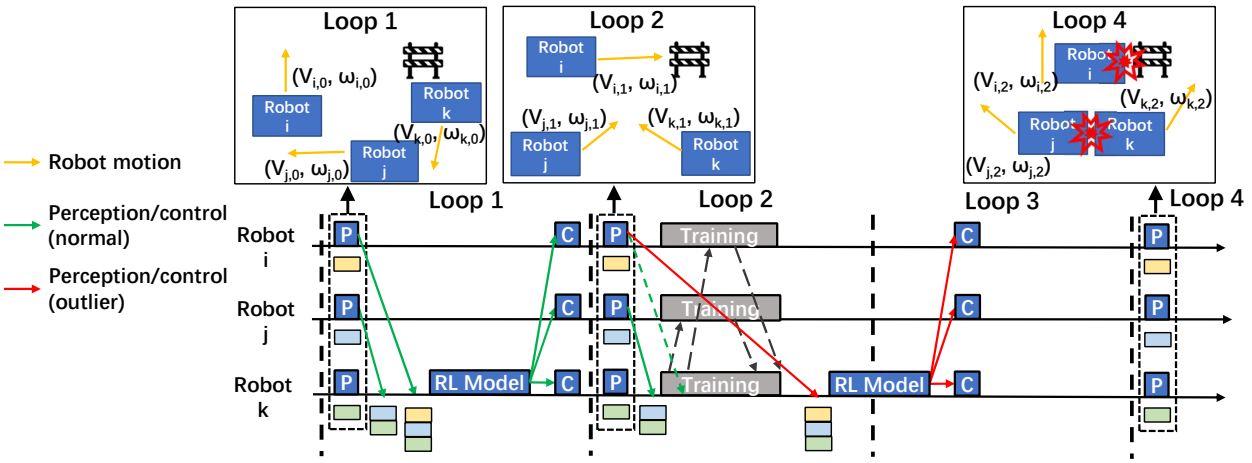
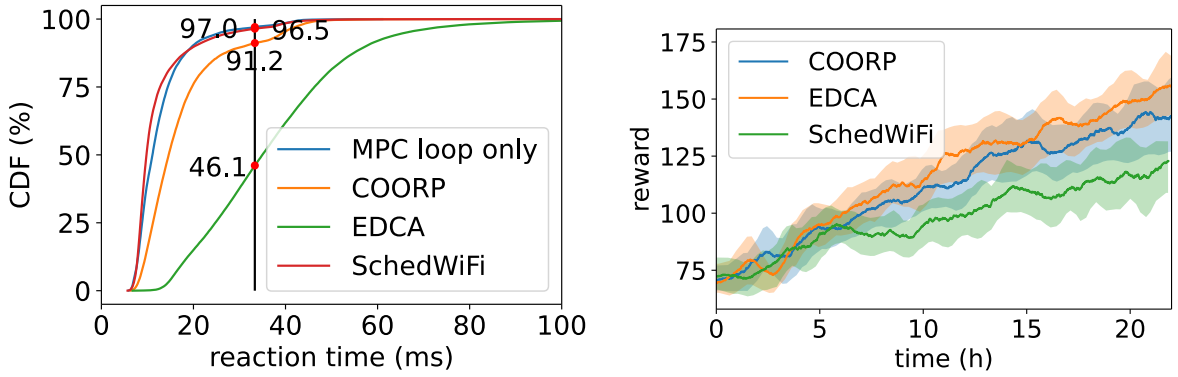


Fig. 1: Workflow of coordinated robotic learning and why fast reaction is important. 'P' stands for 'perception'. 'C' stands for 'control'. Bandwidth-hungry flows for training block the latency-sensitive flow (red line) and cause violation of the reaction time boundary (loop 2), leading to lag and collision (loop 4).

(**R2**): the control model should adapt to a new environment (i.e., converge) as fast as possible.

However, the two coordinations (MPC and DRLC) generate network traffic flows with different characteristics, making it cumbersome for the underlying network supporting systems to meet these two stringent requirements simultaneously. In terms of traffic volume, the messages of MPC (i.e., perception and control messages) are often in small scale (e.g., several KB); while the messages of DRLC (i.e., data dissemination and parameter updates) are often in much larger scale (e.g., tens to hundreds of MB). Still, in terms of traffic pattern, the messages of MPC are generated periodically (e.g., 30Hz) and are latency-sensitive (LS): a high latency would result in violation of reaction time boundary and cause flaws (i.e., violate **R1**); while the messages of DRLC are generated less frequently but are bandwidth-hungry (BH): a limited available bandwidth will slow down the speed that a distributed learned control model adapts to a new environment (i.e., violate **R2**).

Unfortunately, no existing wireless network supporting system meets **R1** and **R2** simultaneously when serving these two aforementioned traffic flows in coordinated robotic learning. Existing systems can be classified into two categories. The first category is *prioritized contention* systems [13]–[17]: each wireless network interface card (WNIC) on each robot occupies the shared wireless channel on demand; WNICs that have high priority packets to transmit are given



(a) Reaction time.

(b) Convergence over time.

Fig. 2: Comparison of different systems. In 'MPC loop only', only the MPC loop process was running. COORP achieved a low percentage of violation of the reaction time boundary (vertical black line) with only a minor slowdown of the training process.

higher chances to occupy the channel before other WNICs. Prioritized contention enables robots to communicate without central control and achieves high bandwidth utilization. However, it fails to achieve **R1** in coordinated robotic learning because multiple WNICs with BH messages to transmit simultaneously contend for the channel, lower the chance for WNICs with LS messages to timely acquire the channel, and still cause latency increase.

The second category is *global planning* systems [18]–[23]: a global scheduler divides the time into time windows and arranges the transmission of LS messages and BH messages into exclusive time windows. Global planning is suitable for cases that a limited available bandwidth does not affect the service quality of the application.

However, in dividing and assigning the time window, global planning has to consider uncertainties from two potential sources: first, the packet transmission time is not stable in wireless networks; second, dividing a time axis across multiple devices requires a global clock but faces clock skewness [18]. In the presence of these uncertainties, global planning faces a paradox to meet both **R1** and **R2** in coordinated robotic learning: on the one hand, as the scheduler cannot collect the uncertainty on each robot in real-time, it has to reserve a large enough time window for each LS message transmission, taking up too much of the bandwidth and violating **R2**; on the other hand, reducing the size of each time window to meet **R2** would make LS transmissions more likely to miss their own windows, violating **R1**.

For instance, we ran a multi-robot navigation coordinated robotic learning application [2] on five robots with SchedWiFi [20], the most relevant global planning system that is designed for low latency of control messages in industrial automation. Despite a low percentage (3.5%) of violation of the reaction time boundary (Figure 2a), SchedWiFi suffered a 49% slowdown of the training process (Figure 2b) due to a 41% reduction of available bandwidth (Table II).

Overall, a system meeting R1 and R2 simultaneously for coordinated robotic learning is highly desired but missing. In this paper, we present COORP, the first network supporting system for cooperating the two types of coordinations (MPC and DRLC) and meeting both **R1** and **R2** for coordinated robotic learning. Lying in the core of COORP is our newly proposed abstraction: *coordinated preemption*. Coordinated preemption should ensure that whenever an LS message from any of the robots is to be transmitted: locally, the WNIC on the robot should immediately serve the LS message for transmission (*local preemption*); globally, the WNIC which serves the LS message should immediately acquire the wireless channel shared by all the WNICs of the robot group (*global preemption*). COORP meets **R1** by enforcing the coordinated preemption. Still, COORP maintains a reasonably high available bandwidth to meet **R2**.

Enforcing *global preemption* is challenging: as multiple WNICs are contending the same wireless channel, when an LS message is to be transmitted by a WNIC, the wireless channel might already be contended by multiple WNICs that are serving a high volume traffic; this lowers the chance that the WNIC with an LS message acquires the channel even with EDCA [17], a technique since WiFi 4 that makes a contention incline to transmissions assigned with higher priority; thus, the LS message is blocked (violate **R1**). To ensure global preemption for coordinated robotic learning, we leverage a key observation that, in the *data dissemination* and *parameter synchronization* of the DRLC, the critical path is dominated by the transmission time of all the BH messages via a single wireless channel. Leveraging this observation, COORP minimizes the contention by only allowing one BH message to be transmitted at any time. By doing so, at any time, there is at most one WNIC with BH messages contending for the channel, and when any LS message is to be transmitted, it can easily win the contention with the highest chance (meet **R1**). Moreover, COORP lets BH messages saturate the wireless channel in a round-robin manner for a high overall bandwidth utilization (meet **R2**).

Enforcing *local preemption* on commodity wireless network interface controllers (WNICs) is confronted with a unique challenge that, in modern WiFi protocol [24], when a large number of packets are being served, newly arrived packets have to wait until the WNIC finishes the

transmission of a number of previous packets due to frame aggregation [25], a key technique in modern WiFi for high bandwidth. Our key observation for this challenge is that, in the MPC of coordinated robotic learning, the LS messages are often generated at a constant frequency which is predictable by COORP. Leveraging this observation, we propose a novel virtual-preemption layer that predicts when an LS message will arrive and limits the number of packets of BH messages the OS passes to the WNIC, such that when an LS message arrives, previous packets buffered by the WNIC have been finished. By doing so, an LS message can be served immediately by the WNIC whenever it is generated (meet **R1**). COORP dynamically estimates the maximum number of packets that could be transmitted before the arrival of the next LS message to meet **R2**. The virtual-preemption layer leverages common operations of the WNIC driver, thus it is portable to different commodity WNICs.

We implemented COORP with around 1000 lines of code in Linux 5.4.84 and ROS2 [26]. We compared COORP with two representative methods, EDCA [17] and SchedWiFi [20], on a typical coordinated robotic learning application [2]. Evaluation shows that:

- COORP achieves fast reaction. COORP realizes a low percentage of violation (8.8%) of reaction time boundary, comparable to the lowest possible case (3.0%) with the same hardware.
- COORP achieves fast model convergence. COORP only incurs 13%~16% slowdown of the training process compared to the fastest baseline (EDCA).
- COORP is easy to use. COORP is integrated with ROS2 and is readily available for robotic applications.

Our major contribution is COORP, the first network supporting system that fulfills both *fast reaction* (**R1**) and *fast model convergence* (**R2**) for mission-critical coordinated robotic learning applications. COORP enables coordinated robotic learning applications to adapt to new environments with minimal time while avoiding lag and collisions among robots. With COORP, more coordinated robotic learning applications can be developed for complex real-world tasks that require close coordination among robots. COORP's code is released on github.com/hku-systems/Coorp.

In the rest of this paper: Section II discusses related work; Section III presents the system model and overview of COORP; Section IV describes the design of COORP; Section V describes the implementation; Section VI shows our evaluation; Section VII concludes.

II. Background and Motivation

A. Coordinated Robotic Learning

Multi-view Perception-Control Coordination. Traditional robot control algorithms mainly leverage local perceptions collected with sensors on each robot [27], [28]. While these algorithms perform well in single-robot tasks, when multiple robots work together, they are unable to leverage perceptions of other robots (e.g., image of an obstacle from different angles) and may lead to suboptimal decisions.

Increasing research efforts in multi-robot applications have led to a multi-view paradigm: multiple robots in a group combine their perceptions of the environment and make coordinated decisions based on the combined perceptions [2], [4], [7]–[9]. For example, in [2], the authors proposed a multi-robot navigation algorithm that combines perceptions of all the robots to get the state of all the obstacles and generates coordinated actions. Without the multi-view paradigm, each robot would only know the position of its nearby obstacles and take suboptimal actions. In real-world deployments, multi-view algorithms may coexist with traditional single-view algorithms for the flexibility of the entire robot group [27].

Distributed Reinforcement Learning Coordination. Deep reinforcement learning trains a deep neural network control model by interacting with the environment, accumulating experiences, and training the control model with the experiences, and is especially effective for robot control involving interaction with complex environments [29], [30]. Despite its flexibility and learning capability, even a well-trained control model requires fine-tuning in new environments [31], [32], which is computation intensive and memory consuming.

Data parallelism is widely adopted in training deep neural networks to alleviate the computation and memory requirements on each single worker. In data parallelism, the training data is partitioned and scattered to multiple workers. These workers train the same control model using their own partition of the data in parallel and synchronize the model parameters with others. There are different paradigms for synchronizing the parameters across workers [33], namely Bulk Synchronous Parallel (BSP), Asynchronous Parallel (ASP), and Stale Synchronous Parallel (SSP). These paradigms differ in when to synchronize the parameters, but they all transmit training data and parameter updates over the network and incur large volume traffic.

B. Related Work

The widely deployed WiFi usually uses distributed contention to decide which WNIC transmits its packets over the shared wireless channel: each time a WNIC wants to transmit, it first sets a counter to a randomly chosen number between 0 and a pre-defined parameter CW; then it decrements the counter when the channel is not used by others until zero before it actually transmits. This requires little coordination among WNICs and is easy to setup, thus it is widely adopted. However, the distributed contention is unable to provide low transmission latency for latency-sensitive traffic. Existing work can be classified into two categories: *prioritized contention* and *global planning*.

Prioritized Contention. This category improves the contention process such that an a WNIC with LS packets to transmit has a higher chance to win in the contention [13]–[17]. For example, EDCA [17], a part of modern WiFi protocol, sets up four traffic categories (`AC_VI`, `AC_VO`, `AC_BE`, and `AC_BK`), and assigns different initial CWs for these traffic categories such that higher priority traffic can gain earlier access to the shared wireless channel. QAir [14] dynamically limits the queue length on each host such that LS packets can bypass most of the packets to be transmitted, but the LS packets still have to wait for packets that are already passed to the WNIC and also contend with transmission from other WNICs. While these systems are effective in daily use cases, they are not suitable for coordinated robotic learning, since multiple robots have BH flows to transmit simultaneously, each trying to saturate the limited bandwidth, causing intensive contention and significantly lowering the chance for LS packets to be transmitted timely.

Global Planning. This category leverages a global scheduler to divide the time into time windows and arrange the transmission of LS messages and BH messages into exclusive time windows [18]–[23]. For example, TDMA-based approaches [19]–[23] proactively assign time windows to different wireless stations. While the contention is fully avoided, they lack flexibility to accommodate dynamically generated traffic like those sent by TCP. Also, they require all the wireless stations to have a global synchronized clock, incurring additional synchronization overhead and are vulnerable to clock skewness [18]. OpenTDMF [18] alleviates the need of a global clock by polling each wireless stations to transmit. It achieves high efficiency and fairness across wireless stations. However, the global scheduler needs to know when an LS message is generated to timely poll the corresponding wireless stations.

C. Motivation

In this work, we propose a new abstraction named *coordinated preemption* consisting of *local preemption* and *global preemption*, and realize it specifically for coordinated robotic learning.

Global Preemption. Global preemption should ensure that, whenever a WNIC has LS messages to transmit, it immediately acquires the wireless channel shared by all the WNICs of the robot group. We observe that, in the data dissemination and parameter synchronization of the DRLC, the critical path is dominated by the transmission time of all the BH messages for synchronization via a single wireless channel. Meanwhile, even a single BH flow is able to saturate the available bandwidth due to its large traffic volume and the limited network capacity. Thus, it is viable to reduce the contention by only allowing a single BH flow to be transmitted at any time, and leverage EDCA to prioritize the transmission of LS flows. In this way, BH flows keep saturating the wireless channel in a round-robin fashion to achieve a high bandwidth utilization (benefitting **R2**), and LS flows only need to contend with a single BH flow at any time (benefitting **R1**). Note that we do not avoid the contention among LS flows, because they have small traffic volume and are less likely to contend with each other.

Local Preemption. Local preemption should ensure that, on each robot, whenever an LS message is to be transmitted, the WNIC on the robot immediately serves the LS message for transmission. Although this was well-studied as a problem of the OS network stack [34]–[41], a pre-condition for existing work to be effective under large traffic volume no longer holds on modern commodity WNICs: *the OS network stack must be the last component where the LS message can be blocked*. Modern commodity WNICs work asynchronously with the OS to achieve high throughput with frame aggregation [25]. Packets delivered to the WNIC accumulate in the WNIC's internal buffer and keep the WNIC busy transmitting. Newly delivered packets have to wait until the ongoing transmission of previous packets is finished, as conceptually depicted in Figure 3. We term this as *local congestion*. Local congestion exists even if the WNIC has multiple hardware queues, because the hardware components for transmission are saturated. Further, local congestion gets amplified when there are BH flows from other robots.

Figure 4 demonstrates local congestion and its amplification. The experiment was carried out with the same testbed as described in Section VI, and FQ-CoDel [34], a queue management algorithm that optimizes for latency-sensitive sparse traffic, was adopted by default such that the LS messages were not blocked in the OS network stack. The LS flow was configured to transmit

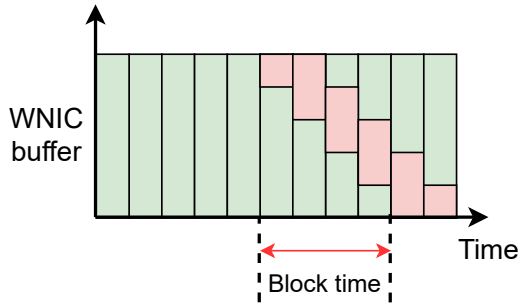


Fig. 3: Conceptual depiction of the congestion in the WNIC. Packets of BH flows (green) delivered by the OS accumulate in the WNIC buffer and keep the WNIC busy transmitting. Packets of LS flows (red) have to wait until transmission of previous packets is finished.

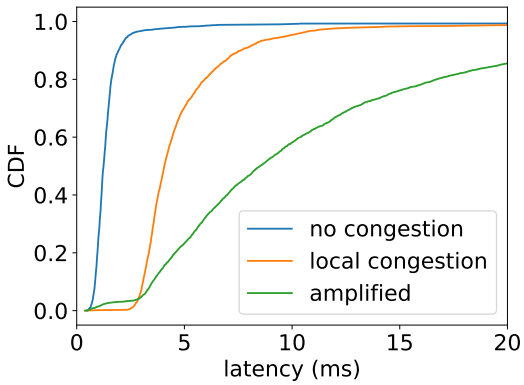


Fig. 4: Local congestion and its amplification. In 'amplified', local congestion is amplified by a BH flow from another robot. Both local congestion and its amplification cause a significant latency increase.

through `AC_VO`, and the BH flow was configured to transmit through `AC_BE`.

In CoorP, local preemption is achieved with a virtual-preemption layer which enables LS flows to virtually preempt local BH flows to avoid the increase of latency. This layer leverages common driver operations (see Section V) and requires software-only modifications to the OS, thus it supports different commodity WNICs.

III. System Overview

A. System Model

In the deployment model of CoorP, two coordinations, i.e., MPC and DRLC as described in Section I, are deployed in a robot group: the MPC makes coordinated decisions for the robot group, and the DRLC adapts the robot group to new environments. Each robot is equipped with sensors (e.g., cameras) that generate perceptions. A leader robot gathers the perceptions of all the robots and holds a deep neural network as the control model to make coordinated decisions. The control model is pre-trained and needs fine-tuning in a new environment for better performance. The other robots (worker robots) are within one-hop distance from the leader robot so that the communication between the worker robots and the leader robot does not need any relay. All robots communicate via a wireless network in a shared channel. RTS/CTS [42] is enabled in

the wireless network to avoid interference caused by hidden terminal problems [43]: multiple worker robots transmit packets simultaneously and interfere with each other.

In the MPC, for the multi-view coordination, the perceptions are generated from the sensors on each robot periodically. Time is divided according to the generation period (e.g., 33ms) of perceptions. Each period is expected to contain exactly one MPC loop. A reaction time boundary no longer than the generation period is configured, and MPC loops are expected to finish within the reaction time boundary. An MPC loop initiated in a period can lag and finish in the next period or later as shown in loop 3 in Figure 1, and put off subsequent MPC loops.

In the DRLC, the leader robot collects a training data item (i.e., perceptions, control messages, and rewards) per MPC loop. The rewards are computed according to subsequent perceptions. When a certain number of training data items are collected, the leader robot disseminates partitions of training data among robots and continues the concurrent MPC loop without collecting training data, because these data items are generated under the old control model. When parameter synchronization finishes and the control model is updated, the leader robot resumes collecting new training data under the updated control model.

B. Overview of COORP

Figure 5 shows the architecture of COORP. COORP fulfills the two requirements of coordinated robotic learning, i.e., *fast reaction* (**R1**) and *fast model convergence* (**R2**), by enforcing coordinated preemption while maintaining a high available bandwidth. The BH controller and global controller enforce global preemption, while the preemption controller and the virtual-preemption layer enforce local preemption.

a) ***Enforcing Global Preemption****:* As discussed in Section II, the concurrently generated BH flows from different robots in the DRLC contend for the channel intensively and increase the latency of LS flows, violating **R1**. We observe that reducing contention intensity is a promising way to enforce **R1** and **R2** simultaneously among flows from different robots. A strawman approach is to leverage a time division multiple access (TDMA) method: dividing time into time slices and exclusively assigning time slices to each BH flow. However, the TDMA method is unsuitable for coordinated robotic learning because different BH flows in the learning process can start and finish transmission at different times due to the variation of training speed and network condition. Once a certain BH flow finishes transmitting, the future time slice assigned to it will be wasted (violating **R2**).

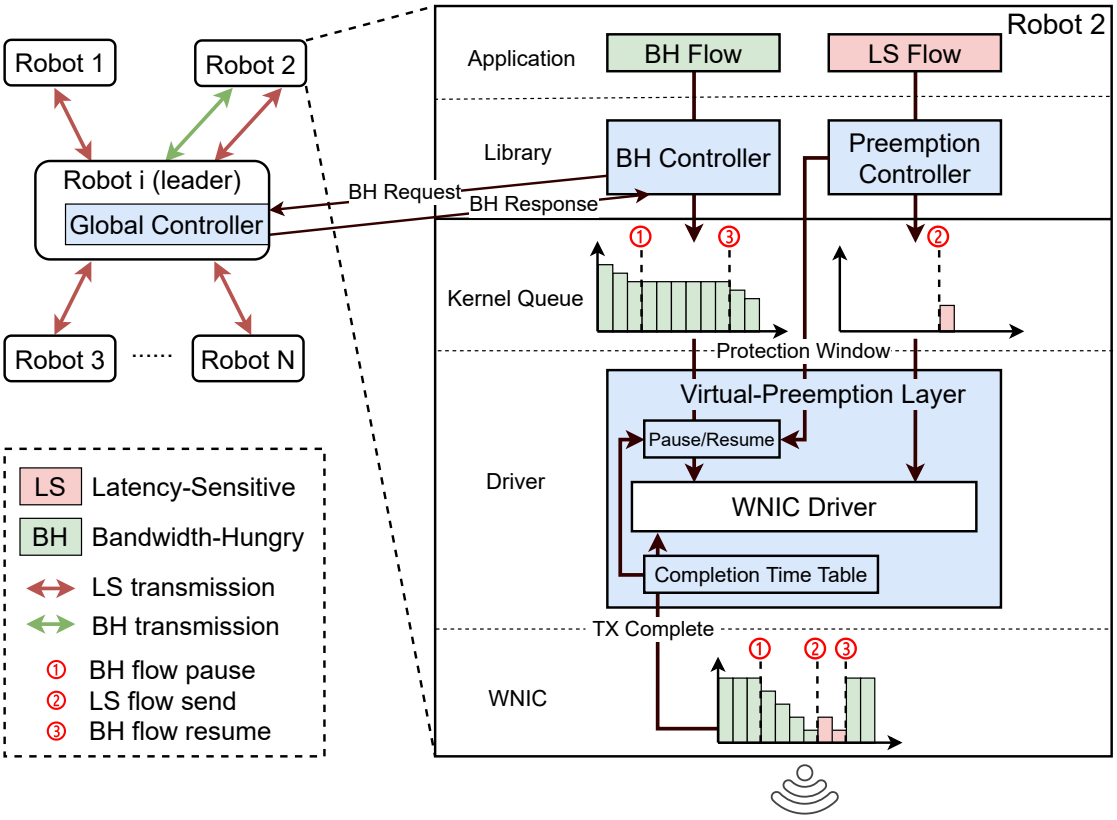


Fig. 5: The architecture of COORP. COORP's components are shaded in blue. ① The virtual-preemption layer pauses the kernel queue of the BH flow to avoid passing too many packets to the WNIC; the WNIC still transmits the packets it has buffered. ② The application sends an LS message; the WNIC serves the LS message immediately because it has finished previous packets passed by the OS. ③ The virtual-preemption layer resumes the kernel queue of the BH flow because the transmission of the LS message is finished. Meanwhile, the global controller and BH controllers ensure there is at most one BH flow being transmitted at any time.

Instead, COORP adopts a request-based semi-TDMA method. When a robot attempts to transmit BH flows, its BH controller will intercept the transmission and request the global controller for permission to transmit. If there is no other BH flow transmitting, the global controller will grant the transmission with a response immediately. If there is some BH flow transmitting, the global controller will postpone the response until the transmitting BH flow finishes. This process only needs a request message and a response message (several bytes), which consumes little of the bandwidth compared to the large size of BH messages (tens to

hundreds of MB), guaranteeing **R2**. Moreover, limiting the number of concurrently transmitting BH flows would not lower bandwidth utilization because a BH flow is able to saturate the bandwidth.

b) ***Enforcing Local Preemption****:* Due to the closed-source design of commodity WNICs, the transmission on the WNIC cannot be manipulated directly. To enforce local preemption, we design and implement a software-only virtual-preemption layer surrounding the WNIC driver between the WNIC and upper kernel layers to virtually enforce preemption for LS flows. The virtual-preemption layer relies on a preemption controller to anticipate the arrival time of LS packets (i.e., the time when an LS flow would send a message). The virtual-preemption layer records the time that BH packets take to be transmitted by the WNIC (transmission completion time) and further estimates the future transmission completion time of new BH packets to be fed to the WNIC. If the transmission completion time of a BH packet would overlap with the arrival time of LS packets, the virtual-preemption layer will stop feeding the BH packet to the WNIC by pausing the kernel queue until the LS packets are sent out. In this way, preemption on the WNIC is virtually achieved and LS packets would not be blocked by the BH packets on the WNIC, minimizing the transmission latency (**R1**). Moreover, the transmission completion time is collected in real-time from the driver locally, which is close to the WNIC. Thus the time to pause kernel queue of BH flows is accurate (benefitting **R1**) and minimal (benefitting **R2**).

IV. Detailed Design

A. Global Controller and BH Controller

In CoorP, a request-based semi-TDMA method as shown in Algorithm 1 is integrated with EDCA enabled, so that contention intensity among BH flows is reduced and LS flows can acquire the channel with the highest probability when contending with BH flows. The global controller on the leader robot maintains a queue Q of robots that requested to transmit BH flows, and a set S of robots that are transmitting BH flows concurrently. In this paper the maximum size of S is set to 1 for the lowest contention intensity.

When a worker robot needs to transmit a BH flow, its BH controller would send a BH request to the global controller on the leader robot and wait for a response (line 13) before transmission. The global controller would allow the requested transmission by sending a BH response when the number of concurrently transmitting BH flows in S has not reached a limit (1 in this paper), so that EDCA can effectively prioritize the transmission of flows according to their latency

Algorithm 1: Request-Based Semi-TDMA

```
1  Function schedule(): /* On global controller */
       Data: Q: queue of robots requesting for transmission; S: set of robots transmitting
2      upon get request from robot r do
3          Q.put(r);
4      upon robot r in S for more than TIMESLICE
5           || get release from robot r do
6          S.remove_if_exist(r);
7      upon r ← Q.get() do
8          wait until |S| < LIMIT;
9          send permit to r;
10         S.add(r);
11 Function send(data): /* On BH controller */
12     while data not all sent do
13         request_wait();
14         transmit data for no more than TIMESLICE;
15         release();
16     end
```

requirements (**R1**). If the limit is reached, the global controller would add the request to Q and postpone the response until the number of concurrently transmitting BH flows decreases (line 8). On receiving the response, the robot keeps transmitting BH flows for a pre-configured time slice (e.g., 5s), or if the robot finishes transmitting BH flows within the pre-configured time, it will notify the global controller to recycle the unused time for other BH flows.

The set S is maintained by the global controller. If the global controller permits a robot to transmit BH flows, the time is recorded and the robot is added to S; If a robot has been transmitting for the pre-configured time according to the permission time records, or a robot notifies that it finishes transmitting BH flows (release), the robot is removed from S.

B. Preemption Controller

Each time an LS flow sends a message, the preemption controller records the time point as a sample. Suppose an LS flow started from time q and its period is p, the ideal arrival time (i.e., when it is generated by the application) of the kth ($k = 0, 1, 2, ...$) message can be modelled as $\hat{T}_k = pk + q$. However, due to random fluctuations, there is a difference between the actual time T_k and the ideal time $\hat{T}_k$. As shown in Figure 6, the fluctuation roughly follows a normal distribution $N(0, \sigma^2)$: $T_k \sim N(pk + q, \sigma^2), k = 0, 1, 2...$ The parameters p, q and σ constitute a time model $\{p, q, \sigma\}$ of an LS flow.

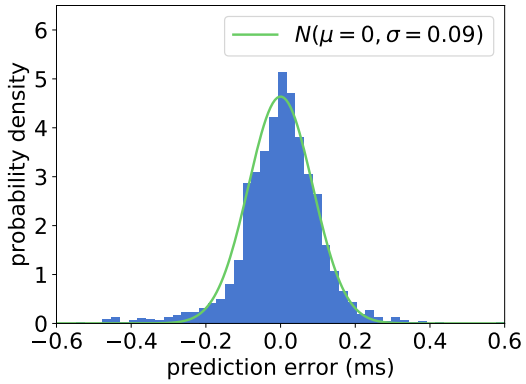


Fig. 6: The prediction error roughly follows normal distribution.

With enough samples of an LS flow, the preemption controller runs linear regression on these samples to estimate the corresponding time model of an LS flow. Since the inferred parameters may become inaccurate over time, the preemption controller recomputes the model in two conditions: 1) the difference between the predicted time and the actual time of a message exceeds 2σ; 2) a statically configured time interval has passed since last recomputing. With the time model $\{p, q, \sigma\}$, the future arrival time T_k of an LS flow can be easily inferred as $[\hat{T}_k - \Delta, \hat{T}_k + \Delta]$ according to normal distribution under a given accuracy requirement denoted as C. We set C to 95% for which the Δ can be approximated with 2σ.

The design of the preemption controller has to cope with two challenges: first, the future arrival time of each LS flow may overlap with each other; second, the future arrival time should be reported to the virtual-preemption layer timely, otherwise there will not be enough time for the WNIC to finish its ongoing transmission. To tackle these challenges, we use a protection window (PW) that covers the future arrival time of multiple LS flows. Suppose the current future

arrival time of all LS flows constitutes a set T and the earliest one is $t = [t_1, t_2]$, we set the PW to t, remove t from T and look up the new earliest one t' in T. If $t' = [t'_1, t'_2]$ overlaps with PW (i.e., $t_1 \leq t'_1 \leq t_2$), t' is combined into PW (i.e., PW= $[t_1, max(t_2, t'_2)]$) and removed from T. The process repeats until a new t' does not overlap with PW, so that PW starting from t covers all future arrival time that needs to be protected as a whole. Upon the ending of a PW, the preemption controller will update the future arrival time of each time model and calculate a new PW, then it notifies the virtual-preemption layer to resume the paused BH flows and update the new PW simultaneously. In this way, the virtual-preemption layer always controls the feeding of BH packets according to the latest PW.

C. Virtual-preemption layer

A WNIC driver mainly contains three major operations: dequeuing packets from upper kernel queues; enqueuing packets to the WNIC; receiving the transmission completion callback of a packet. The virtual-preemption layer wraps these three operations as shown in Algorithm 2.

The virtual-preemption layer maintains a completion time table (CTT) that is used to predict the required time for transmitting specific numbers of packets on the WNIC. The CTT contains rows with form (n, t) which means that it takes time t to complete the transmission of $n + 1$ BH packets on the WNIC. To maintain the CTT, the virtual-preemption layer intercepts into both the enqueue and the transmission completion callback of the WNIC driver (line 11-15 and line 16-20). Each time the driver is enqueueing a BH packet into the WNIC's BH queue, the enqueue time t_{enq} of the packet is recorded, along with the current number of packets n in the WNIC's BH queue. Suppose the packet transmission is completed at time t_{comp}, we learn that the BH queue needs $(t_{comp} - t_{enq})$ time to complete the transmission of a packet together with n packets previously queued in the WNIC queue (line 20). As the completion time varies due to instability and contention in the wireless channel, we keep a number of the most recent records for each n (ranging from 0 to the maximum queue depth of the WNIC), and empirically take the maximum as the estimated transmission time.

Given a PW denoted as $[t_1, t_2]$ and current time T, if $t_1 \geq T$, the virtual-preemption layer protects the LS flows that are supposed to arrive within the PW by intercepting the dequeue operation of the driver (line 1-10). When the driver attempts to dequeue a BH packet from the upper kernel queues, the virtual-preemption layer queries the number of packets currently queued on the WNIC n_{pkt}, and queries the corresponding transmission completion time t_n from the CTT

Algorithm 2: Virtual-Preemption Layer

Data: CTT: completion time table; $[t_1, t_2]$: current PW; N_{samp}: number of samples to keep in the CTT; pkt: BH packet

upon dequeue pkt of kerenl queue **do**
  $n \leftarrow$ current number of packets in hardware queue;
  $t_n \leftarrow max(CTT[n])$;
  $T \leftarrow now()$;
  if $(T < t_1$ && $t_1 - T \leq t_n)$ || $t_1 \leq T \leq t_2$ **then**
    cancel dequeue;
    pause kernel BH queue;
  else
    let the driver dequeue pkt;
  end
upon enqueue pkt of BH flow **do**
  $n \leftarrow$ current number of packets in hardware queue;
  $t_{enq} \leftarrow now()$;
  $pkt.t \leftarrow t_{enq}$;
  $pkt.n \leftarrow n$;
upon complete pkt of BH flow **do**
  $t_{comp} \leftarrow now()$; **if** $|CTT[pkt.n]| > N_{samp}$ **then**
    remove oldest sample from $CTT[pkt.n]$;
  end
  $CTT[pkt.n] \leftarrow CTT[pkt.n] \bigcup \{t_{comp} - pkt.t\}$;
upon receive PW $[t_1', t_2']$ **do**
  $t_1 \leftarrow t_1'$;
  $t_2 \leftarrow t_2'$;
  resume kernel BH queue;

(line 2-3). If $t_n > t_1 - T$ or $t_2 > T > t_1$, the transmission of this BH packet is not likely to be finished before the arrival of the LS flows and would block the transmission of LS flows (line 5). Thus the virtual-preemption layer aborts the dequeuing operation of this packet and further pauses kernel queue of BH flows (line 6-7).

However, if the PW arrives late (e.g., $t_{n-1} > t_1 - T$ when PW is updated), there could be too many BH packets already queued on the WNIC, blocking the incoming LS packets. To avoid this problem, we design the synergy between the preemption controller and the virtual-preemption layer as follows (line 21-24): the controller updates the PW whenever a PW ends (i.e., $t_2 \leq T$) and notifies the virtual-preemption layer upon the update; the virtual-preemption layer first updates its PW to the latest and then resumes the dequeue operation of the BH flows on the driver. In this way, when the driver attempts to dequeue a BH packet, the virtual-preemption layer will always estimate whether the BH packet will block the LS flows on the WNIC according to the latest PW.

The CTT maintained in the virtual-preemption layer guarantees an accurate estimation of transmission completion time. Based on the accurate estimation, the virtual-preemption layer minimizes the time for which the driver stops transmitting BH packets (**R2**) while guaranteeing that BH packets queued on the WNIC will be finished transmitting before LS packets arrive (**R1**).

D. Theoretical Properties of COORP

Here we present two properties of COORP. We define the latency of an LS message as the time between when it arrives at the WNIC and when it is received by another WNIC. First, we denote several latency distributions for the analysis:

D_{free}: the latency of the LS messages when there is no contention and local congestion.

D_{EDCA}: the latency of the LS messages when it is granted the highest priority in EDCA and contends with one BH flow.

D_{cong}: the latency of the LS messages when there is only local congestion.

We also use $D(x)$ to denote the cumulative distribution function (CDF) of D. Note that these latency distributions can be obtained with theoretical analysis [44], [45] or with experiments. The following analysis is independent of how these distributions are obtained.

Property 1. In COORP, at different stages of coordinated robotic learning and on different robots, the latency CDFs of LS messages can follow three cases: $D_{free}(x)$ or $D_{EDCA}(x)$ or

$pD_{free}(x) + (1-p)D_{cong}(x)$, where p is the possibility that the arrival time of LS messages falls in the corresponding PW.

Proof sketch. In COORP, BH flows are controlled by the global controller, and only one robot will be transmitting BH flows at a time with the LIMIT set to one. Note that we omit the influence of congestion and contention among different LS flows because they are small in size. There are three different states from a robot's perspective: 1) no one is transmitting BH flows; 2) the robot itself is transmitting BH flows; 3) another robot is transmitting BH flows. In case 1 and case 3, the CDF is $D_{free}(x)$ and $D_{EDCA}(x)$ respectively by definition. In case 2, due to the protection of COORP, portion p of the messages will get no congestion and follow D_{free}, portion $1-p$ of the messages will get local congestion and follow D_{cong}. Thus the latency CDF is a mixture [46] of $D_{free}(x)$ and $D_{cong}(x)$, with weight p and $1-p$ correspondingly, which is $pD_{free}(x) + (1-p)D_{cong}(x)$.

Property 2. The bandwidth utilization of BH flows in COORP is

$$1 - 2F \times norminv(\frac{1+C}{2}) \tag{1}$$

where $norminv(x)$ is the inverse of the cumulative distribution function for the fluctuations of LS message arrival time which follow $N(0, \sigma^2)$, C is the configured accuracy requirement as was described before, and F is the frequency of LS messages.

Proof sketch. First we determine the length of PW. $P(-\frac{PW}{2} < X < \frac{PW}{2}) = C, X \sim N(0, \sigma^2)$. Thus $P(X < \frac{PW}{2}) = \frac{1+C}{2}$, leading to $PW = 2 \times norminv(\frac{1+C}{2})$.

In COORP, the virtual-preemption layer pauses the BH flow for time PW with frequency F. Thus during each time slice with length L, the time paused totally is $L \times F \times PW$. So the bandwidth utilization is $\frac{L - L \times F \times PW}{L} = 1 - 2F \times norminv(\frac{1+C}{2})$.

V. IMPLEMENTATION

The virtual-preemption layer was implemented as a kernel device module with about 600 lines of code based on Linux 5.4.84 with PREEMP_RT patch [47], and interfaced with the preemption controller through `ioctl()`. Three operations of the WNIC driver was hooked. The operation to enqueue a packet to the WNIC queue was hooked to record the current number of enqueued packets in the WNIC queue and the timestamp. The operation to handle packet completion from the WNIC queue was hooked to compute the transmission completion time, and update the

CTT as was described in Algorithm2. Further, the virtual-preemption layer was hooked into the driver operation of dequeueing packets from mac80211 queues to prevent it from enqueueing too many packets into the WNIC queue. Although the exact code of WNIC drivers differs, these three operations are general and necessary. For evaluation, the hooks were added into the MT76 driver in the kernel source tree, and the functions corresponding to the above three operations were `mt76_queue_ops.tx_queue_skb()`, `mt76_queue_ops.tx_complete_skb()` and `mt76_txq_dequeue()`. The preemption controller was decoupled as two modules, a proxy in the ROS2 rcl (ROS client library) [48] to compute the time model for LS flow, and a controller as a stand-alone ROS2 application. These involved around 400 lines of code in the ROS2 rcl.

Ideally, the BH controller should also be implemented in ROS2. However, ROS2 does not currently provide direct TCP support, and the default UDP support limits the size of the message (64KB) and suffers bandwidth problems in unreliable networks like WiFi [49]. Thus we chose to temporarily implement a message streaming interface with TCP socket, integrated with the BH controller.

VI. Evaluation

Testbed. The evaluation was performed using 5 hosts running Ubuntu 20.04 with Linux 5.4.84 kernel patched with Preempt-RT [50]. Two of the hosts were desktops with Intel Core i7-8700 CPU @ 3.20GHz, and the other three were laptops with different models of CPUs, namely Intel Core i7-10510 CPU@1.80GHz, i5-7200U CPU@2.50GHz, and i7-7700HQ CPU @ 2.80GHz respectively. Robots were simulated with these five hosts, and we refer to these hosts as robots for convenience in this section. One of them (the leader) was equipped with MediaTek MT7612E WNIC and configured an access point on channel 44 (5GHz). The other four were equipped with MediaTek MT7612U USB WNICs and connected to the leader's access point. To measure transmission latency, the hosts were connected to a TP-LINK TL-SG108 desktop Ethernet switch and their system clocks were synchronized with PTP [51]. The synchronization accuracy was reported to be below 10μs by `ptp4l`, sufficient for measuring wireless transmission latency. Note that the accurate time synchronization is only for measuring the latency and reaction time, and is not required by CoorP to work in real deployments.

Baselines. CoorP was compared with two representative methods chosen from the two categories of existing work. For prioritized contention category, EDCA [17] was chosen because it is a part of the IEEE 802.11 standard and is supported by commodity WNICs. For global

planning category, SchedWiFi [20] was chosen because it was specifically designed for periodic latency-sensitive traffic like perceptions of robots.

For EDCA, the LS messages were configured to go through `AC_VO`, the highest priority, and the training data and parameter updates were configured to go through `AC_BE`, lower priority than the LS messages.

For SchedWiFi, since the original paper only evaluated with simulation, to compare its actual performance with CoorP, all its features were implemented on our testbed. ST-Window (STW) is the time window reserved for each LS flow; it was calculated with Equation 2 following the original paper, where L is the size of each message in an LS flow, $SIFS$ is $16\mu s$ according to the standards of IEEE 802.11 [52], and the maximum retransmission time R was set to 7 [53]. TX_{ack} was empirically set to $30\mu s$. The Time-Aware Shaper of SchedWiFi required modification to the PHY layer of the WNICs, which is impractical on commodity WNICs. To achieve the same effect, CoorP's virtual-preemption layer was adapted as SchedWiFi's Time-Aware Shaper which proactively stops the kernel queue before required time point.

$$STW = 2 \times 25\mu s + (\frac{L}{bandwidth} + 2 \times SIFS + TX_{ack}) \times (1 + R) \tag{2}$$

Workload. The multi-robot navigation task in [2] was adopted as the workload, since this task requires multiple robots to gather their real-time sensor data for cooperative control, and is an important building block of mission-critical multi-robot applications such as surveillance and rescue [2], [54], [55]. The task was set up in the Stage simulator [56] with 5 moving obstacles and 5 robots. The deep neural network model and the reward function were the same as that in [2]. Nowadays, the parameter size of real-world reinforcement learning models for robotics has exceeded 100 million [57] and existing methods can reduce the number of parameters by 10x-50x [58]. So the number of parameters was extended to 10 million with dummy data during parameter synchronization to approximate the size of reinforcement learning models in real-world tasks.

To bridge the simulated robots with real-world wireless network, the perception of the robots was retrieved from the simulator and sent to the corresponding host through the wired network. Each robot reported the perception it received to the leader via the wireless network as if the perception was collected from local sensors.

The evaluation focused on these questions:

- RQ1: How is COORP compared to baseline systems in terms of *fast reaction* and *fast model convergence*?
- RQ2: What is the effectiveness of different components of COORP?
- RQ3: How does COORP scale with the number of robots?
- RQ4: What are the limitations of COORP?

A. End-to-end Performance

For end-to-end comparison between COORP and the baseline systems, the reaction time (i.e., the time to finish each MPC loop) and reward were recorded during fine-tuning a pre-trained model. Figure 2a shows the CDF of reaction time of different systems. The vertical line marks the typical reaction time boundary ($\frac{1}{33}$s) [10]–[12]. 'MPC loop only' corresponds to the reaction time when the DRLC was not started and no BH flows were involved, which was the best reaction time any system can achieve with the same hardware and physical layer protocol.

Among all the three systems, EDCA had the highest violation rate (53.9%) of the reaction time boundary, since it did not resolve local congestion in the WNIC, and the simultaneous transmission of multiple (1 to 5) BH flows resulted in heavy contention and amplified local congestion. SchedWiFi best guaranteed fast reaction with a 3.5% violation rate and was very close to 'MPC loop only', because LS messages were assigned exclusive and long transmission time slots according to Equation 2, so as to cope with various uncertainty and avoid interference of any BH messages. COORP also achieved a low violation rate (8.8%) of the reaction time boundary, comparable to SchedWiFi and much lower than EDCA. These results correspond to the recorded CDF of latency of LS messages achieved by different systems in Figure 7.

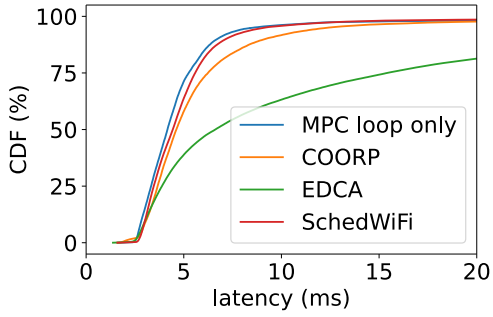


Fig. 7: Latency CDF of LS messages.

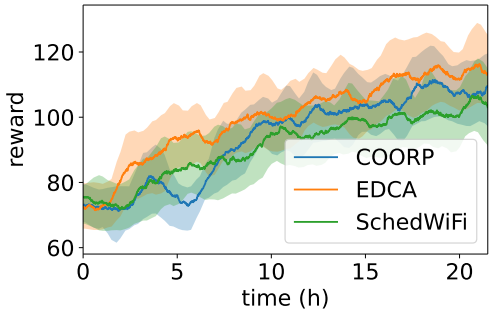


Fig. 8: Convergence over time, SSP.

TABLE I: Time per epoch. Normalized to EDCA's.

	EDCA	SchedWiFi	COORP
BSP	1.00	1.49	1.15
SSP	1.00	1.56	1.13
ASP	1.00	1.56	1.16

The convergence of the control model is shown in Figure 2b (BSP) and Figure 8 (SSP, the limit of version difference of the model between the fastest worker and the slowest worker was set to 1). ASP was not converging in this task using all three systems, thus its figure is omitted. The non-convergence of ASP would be due to outdated updates from slow workers [33]. All three systems started from the same pre-trained model. Despite the jitters caused by the inherent uncertainty of the training process, overall, COORP was slightly slower than EDCA and faster than SchedWiFi. This difference was consistent with time required per epoch shown in Table I: SchedWiFi required 49%~56% more time per epoch, while COORP required 13%~16% more.

TABLE II: Comparison of bandwidth utilization. Normalized to EDCA's.

	EDCA	SchedWiFi	COORP
utilization	1.00 (194.7Mbps)	0.59 (115.1Mbps)	0.86 (168.2Mbps)

Table II shows the bandwidth utilization of different systems which further explains the difference of time per epoch. EDCA achieved the highest throughput (i.e. bytes transmitted per second), and its throughput was treated as the bandwidth of the underlying hardware and physical layer protocol. The bandwidth utilization of SchedWiFi was only 59% since all the robots had to pause BH flows for all the time slots for LS messages, and the sizes of the time slots were often overestimated, limiting the overall time for BH flows to transmit. In COORP, although only one robot was allowed to transmit BH flow at any time, the only transmitting BH flow was able to saturate the bandwidth, and each robot only needed to pause BH flows for its own LS messages. Besides, the coordination between global controller and BH controller incurred little overhead. Thus there was only a minor drop (14%) of the bandwidth utilization, leading to shorter time per epoch and faster model convergence compared with SchedWiFi.

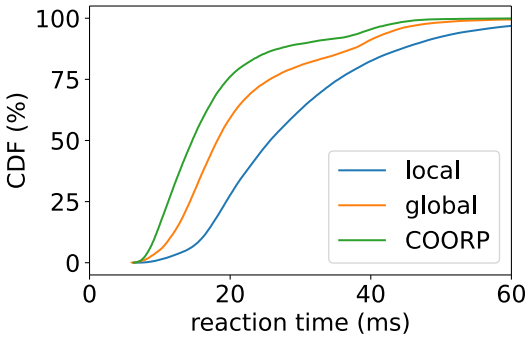


Fig. 9: Contribution of COORP's components to the reaction time. In 'local'('global'), only local(global) preemption was enforced.

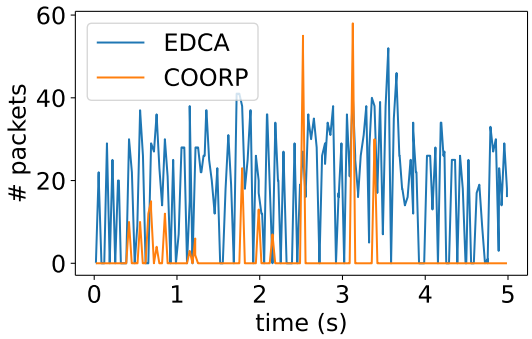


Fig. 10: Effectiveness of the virtual-preemption layer. Y-axis is the number of packets in the WNIC queue when LS packets enqueue, which is zero most of the time in COORP.

TABLE III: Impact of COORP's components to the bandwidth utilization. In 'local'('global'), only local(global) preemption was enabled.

	EDCA	local	global	COORP
utilization	1.00 (194.7Mbps)	0.87 (169.4Mbps)	0.98 (192.1Mbps)	0.86 (168.2Mbps)

B. Effectiveness of Components

To understand the effectiveness of COORP's components, the following two cases were studied: the BH controller was disabled such that only local preemption was enforced; the preemption controller was disabled such that only global preemption was enforced. The corresponding reaction time is shown in Figure 9. When only global preemption was enabled, there was at most one BH flow contending with LS flows at any time, and local congestion was no longer amplified. When only local preemption was enabled, local congestion was eliminated, but LS flows still contend with multiple BH flows. Thus global preemption was more effective when enabled alone, but the combination of them led to even lower reaction time.

The corresponding bandwidth utilization is shown in Table III. Global preemption sacrificed little bandwidth because each single BH flow was able to saturate the bandwidth. Local preemption accounted for most of the sacrifice of the bandwidth because it had to pause ongoing BH flow in each protection window.

To further validate the effectiveness of the virtual-preemption layer, the number of BH packets queued in the WNIC when LS packets were passed to the WNIC was inspected, shown in Figure 10. In EDCA, BH packets accumulated in the WNIC queue and would block the transmission of LS packets. In CoorP, the virtual-preemption layer avoided passing too many BH packets to the WNIC before LS packets. Thus, the WNIC was able to finish transmitting BH packets before LS packets were passed and transmit LS packets as soon as possible. However, the CTT of the virtual-preemption layer was not always accurate, thus it was not able to ensure the WNIC queue was always clean.

C. Scalability to the Number of Robots

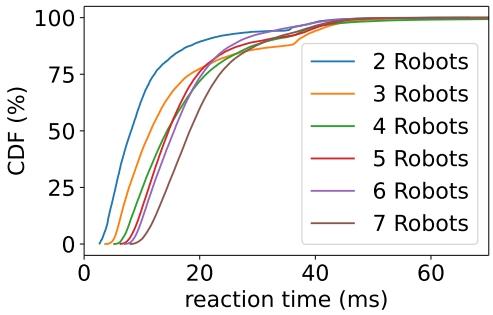


Fig. 11: Scalability of reaction time.

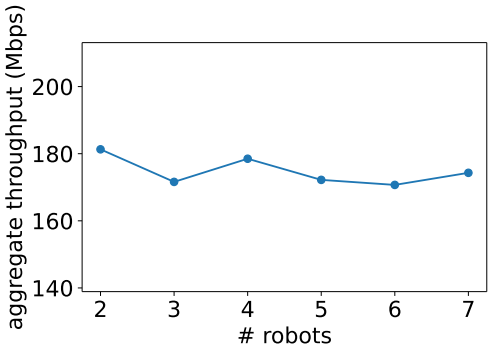


Fig. 12: Scalability of bandwidth utilization.

Figure 11 shows the scalability of CoorP in terms of reaction time. With more robots, it takes more time to gather all the perceptions from all the robots, thus there was a slight increase. Figure 12 shows the scalability of bandwidth utilization. In CoorP, the overhead for reducing contention comes from the requests and responses between BH controller and global controller, which is small compared to the large traffic volume of BH flows. The preemption requires each robot only to pause for its own LS messages, which is independent of the number of robots. Thus the bandwidth utilization scaled well and the variation mainly came from random jitters.

D. Performance under Low Available Bandwidth

In coordinated robotic learning, robots move around and may result in a drop in available bandwidth. To understand the performance of CoorP under such scenarios, two individual cases were set up, and the results were shown in Figure 13 and Table IV.

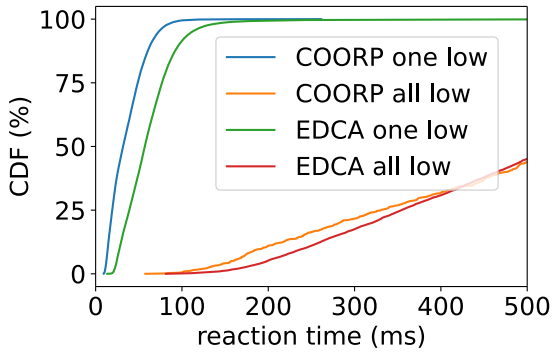


Fig. 13: Reaction time under low available bandwidth. In 'one(all) low', one(all) of the robots used the lowest transmission rate.

TABLE IV: Throughput under low available bandwidth. In 'one(all) low', one(all) of the robots used the lowest transmission rate.

	EDCA	COORP
one low	99.8Mbps	148.1Mbps
all low	19.6Mbps	16.3Mbps

In the first case, denoted as 'one low', one of the robots was configured to use the lowest transmission rate (or bitrate) `vht-mcs-5 1:0` using `iw` tool. Compared to COORP, EDCA achieved a lower throughput. This would be related to the performance anomaly of 802.11 previously discussed by [37], [59] that the stations in a wireless network would achieve similar throughput even when their network conditions differ. By only allowing one robot to transmit simultaneously, COORP enabled each robot to achieve the highest throughput allowed by its network condition, resulting in higher aggregate throughput (averaged over time). This indicates that COORP provides faster training than EDCA when the available bandwidth of a small number of robots is low.

In the second case, denoted as 'all low', all the robots were configured to use the lowest transmission rate. The throughput was restricted by the physical connection, thus both EDCA and COORP suffered a low throughput which would inevitably slow down the training. The low transmission rate also resulted in significant increase of reaction time shown in Figure 13. This would be due to the large gap between the traffic volume and the available bandwidth.

E. Limitations

COORP has the following limitations. First, COORP focused on the case of a single robot group and did not handle the case when multiple robot groups coexist and use the same wireless channel. We leave this as our future work. Second, instead of real robots, COORP was evaluated on a semi-simulated testbed where the interaction between the robots and the environment was simulated while the communication among the robots went through a real-world wireless network. Since this paper focused on the network supporting systems, such settings would be sufficient for the evaluation purpose.

VII. CONCLUSION

In this paper, we identify two technical requirements of the network supporting system for mission-ciritcal coordinated robotic learning, namely fast reaction and fast model convergence, and present COORP, the first network supporting system that achieves both of the requirements simultaneously. COORP enables robot groups to adapt to new environments with minimal time consumption while avoiding lag and collisions. COORP's code is released on github.com/hku-systems/Coorp.

REFERENCES

[1] J. P. Queralta, J. Taipalmaa, B. Can Pullinen, V. K. Sarker, T. Nguyen Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," vol. 8, pp. 191 617–191 643, publisher: IEEE.

[2] R. Han, S. Chen, and Q. Hao, "Cooperative Multi-Robot Navigation in Dynamic Environment with Deep Reinforcement Learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 448–454, iSSN: 2577-087X.

[3] E. Vidal, J. D. Hernández, N. Palomeras, and M. Carreras, "Online robotic exploration for autonomous underwater vehicles in unstructured environments," in *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, pp. 1–4.

[4] Y. Zhang, D. Shi, Y. Wu, Y. Zhang, L. Wang, and F. She, "Networked Multi-robot Collaboration in Cooperative–Competitive Scenarios Under Communication Interference," in *Collaborative Computing: Networking, Applications and Worksharing*, H. Gao, X. Wang, M. Iqbal, Y. Yin, J. Yin, and N. Gu, Eds. Cham: Springer International Publishing, 2021, vol. 349, pp. 601–619.

[5] R. Nishihara, P. Moritz, S. Wang, A. Tumanov, W. Paul, J. Schleier-Smith, R. Liaw, M. Niknami, M. I. Jordan, and I. Stoica, "Real-Time Machine Learning: The Missing Pieces," *arXiv:1703.03924 [cs]*, May 2017, arXiv: 1703.03924. [Online]. Available: http://arxiv.org/abs/1703.03924

[6] P. Caloud, Wonyun Choi, J. Latombe, C. L. Pape, and M. Yim, "Indoor automation with many mobile robots," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pp. 67–72 vol.1.

[7] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous Robots and Agents*, ser. Studies in Computational Intelligence, S. C. Mukhopadhyay and G. S. Gupta, Eds. Springer, pp. 9–16.

[8] X. Kong, B. Xin, F. Liu, and Y. Wang, "Revisiting the master-slave architecture in multi-agent deep reinforcement learning." [Online]. Available: http://arxiv.org/abs/1712.07305

[9] K. Zhang, Z. Yang, and T. Basar, "Networked multi-agent reinforcement learning in continuous spaces," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 2771–2776.

[10] H. Nascimento, M. Mujica, and M. Benoussaad, "Collision avoidance interaction between human and a hidden robot based on kinect and robot data fusion," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 88–94, 2021, publisher: IEEE. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03139189

[11] X. Sun, X. Weng, and K. Kitani, "When We First Met: Visual-Inertial Person Localization for Co-Robot Rendezvous," *arXiv:2006.09959 [cs]*, Nov. 2020, arXiv: 2006.09959. [Online]. Available: http://arxiv.org/abs/2006.09959

[12] L. S. Scimmi, M. Melchiorre, S. Mauro, and S. Pastorelli, "Experimental Real-Time Setup for Vision Driven Hand-Over with a Collaborative Robot," in *2019 International Conference on Control, Automation and Diagnosis (ICCAD)*, Jul. 2019, pp. 1–5.

[13] D. Kim, I. Yeom, and T.-J. Lee, "Mitigating tail latency in IEEE 802.11–based networks," *International Journal of Communication Systems*, vol. 31, no. 1, p. e3404, 2018, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.3404.

[14] Changhua Pei, Youjian Zhao, Yunxin Liu, Kun Tan, Jiansong Zhang, Yuan Meng, and Dan Pei, "Latency-based WiFi congestion control in the air for dense WiFi networks," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, Jun. 2017, pp. 1–10.

[15] M. Heusse, F. Rousseau, R. Guillier, and A. Duda, "Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs," p. 12.

[16] A. Saeed, M. Ammar, E. Zegura, and K. Harras, "If you can't Beat Them, Augment Them: Improving Local WiFi with Only Above-Driver Changes," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, Sep. 2018, pp. 378–388, iSSN: 1092-1648.

[17] "IEEE 802.11e-2005," Feb. 2021, page Version ID: 1006381497. [Online]. Available: https://en.wikipedia.org/w/index.php?title=IEEE_802.11e-2005&oldid=1006381497

[18] Z. Yang, J. Zhang, K. Tan, Q. Zhang, and Y. Zhang, "Enabling TDMA for today's wireless LANs," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2015, pp. 1436–1444, iSSN: 0743-166X.

[19] Y. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka, "RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications," in *2013 IEEE 34th Real-Time Systems Symposium*, Dec. 2013, pp. 140–149, iSSN: 1052-8725.

[20] G. Patti, G. Alderisi, and L. L. Bello, "SchedWiFi: An innovative approach to support scheduled traffic in ad-hoc industrial IEEE 802.11 networks," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sep. 2015, pp. 1–9, iSSN: 1946-0759.

[21] F. Santos, L. Almeida, and L. S. Lopes, "Self-configuration of an adaptive TDMA wireless communication protocol for teams of mobile robots," in *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Sep. 2008, pp. 1197–1204, iSSN: 1946-0759.

[22] S. Cao and V. C. S. Lee, "A Novel Adaptive TDMA-Based MAC Protocol for VANETs," *IEEE Communications Letters*, vol. 22, no. 3, pp. 614–617, Mar. 2018, conference Name: IEEE Communications Letters.

[23] X. Guo, S. Wang, H. Zhou, J. Xu, Y. Ling, and J. Cui, "Performance Evaluation of the Networks with Wi-Fi based TDMA Coexisting with CSMA/CA," *Wireless Personal Communications*, vol. 114, no. 2, pp. 1763–1783, Sep. 2020.

[24] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, Feb. 2021, conference Name: IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016).

[25] "Frame aggregation," Sep. 2020, page Version ID: 978836459. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Frame_aggregation&oldid=978836459

[26] "ROS Documentation." [Online]. Available: https://docs.ros.org/

[27] J. Scherer, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, H. Hellwagner, and B. Rinner, "An Autonomous Multi-UAV System for Search and Rescue," in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. Florence Italy: ACM, May 2015, pp. 33–38.

[28] R. N. Darmanin and M. K. Bugeja, "A review on multi-robot systems categorised by application domain," in *2017 25th Mediterranean Conference on Control and Automation (MED)*, Jul. 2017, pp. 701–706, iSSN: 2473-3504.

[29] M. J. Matarić, "Reinforcement Learning in the Multi-Robot Domain," in *Robot Colonies*, R. C. Arkin and G. A. Bekey, Eds. Boston, MA: Springer US, 1997, pp. 73–83.

[30] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, Apr. 2021, publisher: SAGE Publications Ltd STM.

[31] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "RoboNet: Large-Scale Multi-Robot Learning," *arXiv:1910.11215 [cs]*, Jan. 2020, arXiv: 1910.11215. [Online]. Available: http://arxiv.org/abs/1910.11215

[32] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning," *arXiv:1803.11347 [cs, stat]*, Feb. 2019, arXiv: 1803.11347.

[33] X. Zhao, A. An, J. Liu, and B. X. Chen, "Dynamic Stale Synchronous Parallel Distributed Training for Deep Learning," *arXiv:1908.11848 [cs, stat]*, Aug. 2019, arXiv: 1908.11848. [Online]. Available: http://arxiv.org/abs/1908.11848

[34] D. Taht, J. Gettys, E. Dumazet, T. Hoeiland-Joergensen, E. Dumazet, P. McKenney, J. Gettys, T. Hoeiland-Joergensen, and P. McKenney, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm." [Online]. Available: https://tools.ietf.org/html/rfc8290

[35] S. Seytnazarov and Y. Kim, "QoS-aware adaptive MPDU aggregation of VoIP traffic on IEEE 802.11n WLANs," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov. 2014, pp. 356–359, iSSN: 2165-963X.

[36] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, "The Good, the Bad and the WiFi: Modern AQMs in a residential setting," *Computer Networks*, vol. 89, pp. 90–106, Oct. 2015.

[37] T. Høiland-Jørgensen, M. Kazior, D. Täht, A. Brunstrom, and P. Hurtig, "Ending the Anomaly: Achieving Low Latency and Airtime Fairness in WiFi," p. 15, 2017.

[38] C. A. Grazia, "Towards 1 ms WLAN Latency," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2018, pp. 278–284, iSSN: 2160-4886.

[39] A. Showail and B. Shihada, "Battling Latency in Modern Wireless Networks," *IEEE Access*, vol. 6, pp. 26 131–26 143, 2018, conference Name: IEEE Access.

[40] M. Backhaus, M. Theil, M. Rossberg, and G. Schaefer, "Towards a Flexible User-Space Architecture for High-Performance IEEE 802.11 Processing," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2018, pp. 1–9, iSSN: 2160-4886.

[41] D. Tardioli, R. Parasuraman, and P. Ögren, "Pound: A multi-master ROS node for reducing delay and jitter in wireless multi-robot networks," *Robotics and Autonomous Systems*, vol. 111, pp. 73–87, Jan. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017309144

[42] "IEEE 802.11 RTS/CTS," Nov. 2020, page Version ID: 991386765. [Online]. Available: https://en.wikipedia.org/w/index.php?title=IEEE_802.11_RTS/CTS&oldid=991386765

[43] "Hidden node problem," Oct. 2020, page Version ID: 984728511. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hidden_node_problem&oldid=984728511

[44] P. Schulz, L. Ong, P. Littlewood, B. Abdullah, M. Simsek, and G. Fettweis, "End-to-End Latency Analysis in Wireless Networks with Queuing Models for General Prioritized Traffic," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2019, pp. 1–6, iSSN: 2474-9133.

[45] K. Medepalli and F. A. Tobagi, "On Optimization of CSMA/CA based Wireless LANs: Part I - Impact of Exponential Backoff," in *2006 IEEE International Conference on Communications*, vol. 5, Jun. 2006, pp. 2089–2094, iSSN: 1938-1883.

[46] "Mixture distribution," Mar. 2021, page Version ID: 1013174298. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mixture_distribution&oldid=1013174298

[47] RTwiki. [Online]. Available: https://rt.wiki.kernel.org/index.php/Main_Page

[48] "ros2/rcl," Apr. 2021, original-date: 2015-02-21T00:06:40Z. [Online]. Available: https://github.com/ros2/rcl

[49] "DDS tuning information." [Online]. Available: https://index.ros.org/doc/ros2/Troubleshooting/DDS-tuning/

[50] "realtime:start [Wiki]." [Online]. Available: https://wiki.linuxfoundation.org/realtime/start

[51] "Precision Time Protocol," Jan. 2021, page Version ID: 1001872854. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Precision_Time_Protocol&oldid=1001872854

[52] "Short interframe space," page Version ID: 988080746. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Short_Interframe_Space&oldid=988080746

[53] "802.11/MAC/Lower/Retransmissions – WARP Project." [Online]. Available: https://warpproject.org/trac/wiki/802.11/MAC/Lower/Retransmissions

[54] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," *arXiv:1709.10082 [cs]*, May 2018, arXiv: 1709.10082. [Online]. Available: http://arxiv.org/abs/1709.10082

[55] A. Benevento, M. Santos, G. Notarstefano, K. Paynabar, M. Bloch, and M. Egerstedt, "Multi-Robot Coordination for Estimation and Coverage of Unknown Spatial Fields," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 7740–7746, iSSN: 2577-087X.

[56] R. Vaughan, "rtv/Stage," May 2021, original-date: 2010-05-19T13:40:02Z. [Online]. Available: https://github.com/rtv/Stage

[57] "RoboNet: A Dataset for Large-Scale Multi-Robot Learning – The Berkeley Artificial Intelligence Research Blog." [Online]. Available: https://bair.berkeley.edu/blog/2019/11/26/robo-net/

[58] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "AutoCompress: An automatic DNN structured pruning framework for ultra-high compression rates," vol. 34, no. 4, pp. 4876–4883, number: 04. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5924

[59] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 2, Mar. 2003, pp. 836–843 vol.2, iSSN: 0743-166X.