Resilient Time Synchronisation for Aerial Swarms by Distributed Graph Neural Networks

Yan Zong, *Member, IEEE,* Lejun Chen, *Senior Member, IEEE,*
Pep Canyelles-Pericas, *Member, IEEE,* Ningyun Lu, *Member, IEEE,* Bin Jiang, *Fellow, IEEE*

***Abstract*—Aerial swarms consisting of multiple Unmanned Aerial Vehicles (UAVs) have been applied across various domains. Time synchronisation is important for swarm wireless networks. However, most studies assume that UAV clocks are synchronised, without addressing how synchronisation is achieved. Moreover, existing control techniques are typically designed in a centralised manner with a known topology, limiting their applicability to large swarms or those with dynamic wireless networks caused by high mobility or communication link failures resulting from jamming attacks. These limitations may lead to the instability of the controllers, or even the downtime of the entire swarm system, particularly under the communication link failures. Therefore, in this work, we propose leveraging Graph Neural Networks (GNNs) to achieve resilient clock synchronisation among UAVs. First, we integrate the heat kernel into the graph neural network, allowing it to retain low-frequency graph signals while attenuating high-frequency components. This is consistent with the aim of time synchronisation, which is to ensure that the states of all the clocks are the same, corresponding to low-frequency graph signals. Meanwhile, we introduce a distributed GNN architecture with low communication overhead, in contrast to existing decentralised GNNs that rely on fully-connected networks. Through adversarial imitation learning, our GNN-based control policies achieve similar synchronisation performance without requiring re-training when scaled to large swarms, as compared to the centralised controller using fully-connected wireless networks. Once trained, the proposed GNN-based control policies are also resilient to varying wireless networks, including temporary or permanent communication link failures, and can maintain synchronisation even when the swarm is split into two disconnected parts.**

***Index Terms*—Graph neural networks, time synchronisation, aerial swarms.**

I. Introduction

RECENTLY, aerial swarms, which consist of multiple Unmanned Aerial Vehicles (UAVs), have been utilised in many domains, including logistics, resource distribution and agriculture. Flying Ad-hoc NETworks (FANETs) are wireless communication networks in which all UAVs (also referred to as vehicles or nodes) communicate directly with each other, without any communication links to a ground station [1]. Time synchronisation is crucial for enabling data fusion, distance measurements and collaborative control in such wireless networks [2]. However, most existing studies implicitly assume that all UAV clocks are synchronised, without explicitly addressing how such synchronisation is achieved in the networks [2], [3]. Due to the high mobility of UAVs in aerial swarms (often exceeding $30\,\mathrm{km\,h^{-1}}$), the wireless network changes continuously [1], [2], [5]. In addition, communication networks are susceptible to temporary or permanent wireless communication link failures caused by jamming attacks[1], further contributing to the dynamic swarm network variations. Thus, achieving time synchronisation in a mobile and dynamic swarm communication network remains a non-trivial issue, and without synchronisation, the swarm system may fail. In this work, we aim to use Graph Neural Networks (GNNs) to achieve resilient clock synchronisation, thereby addressing the assumptions and challenges of synchronisation in swarms.

Time synchronisation has been studied in the control engineering community under the topic of multi-agent systems. However, existing control strategies (e.g. [7], [8]) are typically designed based on known topologies in a centralised manner, limiting their applicability to swarm systems with large-scale or dynamically changing wireless networks. The transferable [9] and decentralised [4] characteristics of graph neural networks offer a promising solution to these challenges. Nevertheless, many GNN-based control policies are still evaluated in centralised simulations. Although [10] implements GNNs in a decentralised form, it adopts a fully-connected wireless network among five vehicles, which could result in significant communication overhead when applied to a large number of UAVs.

Therefore, we propose distributed graph neural networks to achieve clock synchronisation in aerial swarms, thereby addressing the challenges of scalability and implementation. Furthermore, we demonstrate that our trained GNN models are resilient to varying wireless networks, including the temporary or permanent communication link failures caused by jamming

Manuscript received November 26, 2024; revised May 22, 2025; accepted June 18, 2025. This work was supported in part by the National Natural Science Foundation of China (NSFC) under grants 62303221 and 62333011, and in part by the Fundamental Research Funds for the Central Universities under grant NS2024020. *(Corresponding author: Bin Jiang.)*

Y. Zong is with the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China, and also with the Purple Mountain Laboratories, Nanjing, China (e-mail: y.zong@nuaa.edu.cn).

L. Chen, N. Lu and B. Jiang are with the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China (e-mail: {l.chen, luningyun, binjiang}@nuaa.edu.cn).

P. Canyelles-Pericas is with the Department of Robotics and Mechatronics, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, the Netherlands (e-mail: {j.canyellespericas@utwente.nl).

[1]We consider jamming attacks in which a malicious jammer broadcasts a high-power signal [6]. This not only disrupts packet reception by introducing strong interference during data transmission, but also prevents UAVs from accessing the wireless channel by continuously occupying it. In other words, jamming attacks targeting the communication links between UAVs can lead to temporary or permanent communication failures on the affected links (i.e. no wireless packets can be successfully received on those links), thereby changing the network topology. Attacks involving adversarial data injection into wireless packets or partial data reception by UAVs are beyond the scope of this work.

TABLE I
COMPARISON OF EXISTING PROTOCOLS.

Community	Protocol	Key algorithm	Task	One-hop or network-wide	Resilience to topological changes under link failures
Communications	[2]	Recursive averaging	Synchronisation	One-hop	✗
	PkCOs [12]	Proportional-integral control		One-hop	✗
	BATS [14]	Least squares regression		One-hop	✗
Control theory	R-PkCOs [8]	Robust control		One-hop	✗
	[15]	Maximum likelihood estimation		One-hop	✗
	PISync [16]	Adaptive control		One-hop	✗
Machine learning	GCNNs and GRNNs [31]	Graph neural network	Flocking	Network-wide	✗
	[47]	Graph neural network		Network-wide	✗

attacks, and can maintain synchronisation even when the swarm is split into two parts. Such a GNN-based control policy may serve as a backup controller in safety-critical applications [11].

A. Related Work

We begin by analysing time synchronisation from the perspectives of communication and control. Next, we present graph neural networks in both the spectral and spatial domains, followed by their application in closed-loop systems. Finally, we discuss the resilience of GNN-based control policies under varying wireless networks, with a particular focus on the temporary or permanent communication link failures induced by jamming attacks.

In the literature, the Global Navigation Satellite System (GNSS) is widely used as a reference clock to synchronise all UAV clocks in an aerial swarm. Specifically, each UAV is equipped with a GNSS receiver, enabling time synchronisation across the swarm [5]. However, GNSS signals are not always available, such as in indoor environments or forested areas [1].

1) Time Synchronisation from Communication and Control Perspectives: Owing to the significance of time synchronisation, it has been extensively studied in both the wireless communications and control systems communities. The wireless communications community has proposed packet exchange-based algorithms (e.g. [2], [12]–[14]). In these approaches, timestamps are locally generated by each clock and wirelessly transmitted. Upon receiving packets containing these timestamps, the clock offset, which is the difference between UAV clocks, is estimated and used for clock adjustment.

In packet exchange-based synchronisation methods, early studies mainly focused on designing packet-exchange strategies to accurately measure clock offsets, thus achieving time synchronisation by applying correction values (obtained from the offsets) to the clocks. Later, attention shifted to employing advanced processing techniques, such as recursive averaging [2], least squares regression [14] and maximum likelihood estimation [15], to further improve the precision of the correction values. As shown in Table I, the above studies only investigate one-hop time synchronisation performance, while overlooking synchronisation from a network-wide perspective.

For the control systems community, clock synchronisation is typically studied under the topic of multi-agent systems. Numerous control strategies, including dynamic control [7], robust control [8] and adaptive control [16], have been proposed to achieve precise time synchronisation. However, the parameters of these strategies are usually designed based on pre-known network topologies in a centralised manner, which limits their applicability in swarm systems with large-scale structures or communication networks subject to topological changes caused by the high mobility or the communication link failures. Inspired by the transferable nature of graph neural networks, this study proposes leveraging GNNs to achieve resilient time synchronisation in aerial swarms with large-scale or dynamically changing networks, thereby addressing the challenges of scalability and implementation [4].

2) Graph Neural Network-based Control Policies: There are two types of graph neural networks: spectral and spatial GNNs. Graph neural network inference includes two phases: the first phase involves aggregating and combining node or edge features, while the second phase is pooling or readout. The pooling/readout phase depends on the type of task, such as node-level, edge-level, or graph-level tasks [17]–[19]. For instance, the output from the first phase may be directly used for node-level tasks, while a MultiLayer Perceptron (MLP) is applied to handle graph-level tasks [17].

For spectral GNNs, a spectral graph signal (obtained from a spatial graph signal via the graph Fourier transform) is first processed using a spectral filter (e.g. low-pass, band-pass, or high-pass). Then, the inverse graph Fourier transform is applied to convert the processed spectral signal back to the spatial domain. The (inverse) graph Fourier transform requires computing the eigendecomposition of the graph Laplacian matrix, which demands intense computation, especially for large graphs (i.e. large-scale FANETs). To address this issue, several studies propose using Chebyshev expansion [20] or Jacobi basis [21] functions to approximate the spectral filter, thereby eliminating the need to calculate the eigenvectors of the Laplacian matrix. Moreover, the work in [22] introduces the Graph Convolutional Network (GCN) using a first-order Chebyshev polynomial. In other words, each layer of the GCN aggregates features from one-hop neighbours, which are then used to calculate a new feature vector for the next layer.

Practically, reaching an isothermal equilibrium in the heat diffusion process is similar to realising time synchronisation in aerial swarms [23]. In the heat diffusion process, heat flows through the edges of a graph, and the diffusion rate is proportional to both the temperature differences between

adjacent nodes and the edge weights [24]. The heat kernel [25] has been used as a solution to the heat diffusion problem. In the literature (e.g. [24], [26], [27]), heat kernel-based GNNs have also been proposed. The heat kernel can be regarded as a low-pass filter, which smooths out the features of all nodes in a graph and retains only the low-frequency graph signals[2] [24], [26], [28]. It explicitly attenuates high-frequency graph signals [26], which is consistent with the aim of time synchronisation, namely, letting the clock offsets of all UAVs converge to zero (i.e. a low-frequency graph signal). Hence, in this work, we are interested in leveraging heat kernel-based GNNs to study time synchronisation in aerial swarms.

On the other hand, spatial GNNs usually define operations by leveraging graph structural information, including node and edge features. These operations are designed to aggregate features from neighbouring nodes, which is also referred to as the message passing scheme [17], [29]. Due to this feature aggregation mechanism, the GCN also belongs to the spatial GNN category. Note that stacking ℓ GNN layers is equivalent to aggregating (hidden) features from ℓ-hop neighbours [18], [29].

In addition to node-level classification [22], [30] and graph-level classification [20], graph neural networks have also recently been employed in closed-loop systems, such as flocking [10], [31] and path planning [32]. The first phase of the GNNs in [4], [10], [31] involves polynomial operations on a graph matrix (e.g. a normalised adjacency matrix), while the second phase is an MLP-based readout. However, the above GNN-based control policies are conducted in centralised simulations. The features of all vehicles are sent to a central vehicle for processing via polynomial operations and an MLP, and the processed data is then returned to each vehicle. Overall, these GNNs fall into the category of centralised control strategies [10], [33]. Even though [10] implements a GNN in a decentralised form, it relies on a fully-connected wireless network for communication. Such a decentralised GNN increases communication overhead when scaled to a large number of vehicles. To address this, we introduce distributed graph neural networks, which avoid significant communication overhead in aerial swarms, for time synchronisation in wireless networks.

Aerial swarms are particularly vulnerable to jamming attacks [6], [34], as wireless communication links among UAVs can be easily disrupted by various types of jamming, such as constant [35], reactive [36], deceptive, and random and periodic jamming [37]. The main differences among these attacks lie in the specific strategies used by the malicious jammer to transmit interfering signals. Nevertheless, the effects of such attacks remain similar and can, without loss of generality, be represented as the temporary or permanent communication link failures, which in turn lead to the changes in the wireless network topology. Such topological variations may further result in the instability of time synchronisation systems in aerial swarms, particularly when control strategies are fixed and designed based on known network topologies. Moreover, an attack on the central UAV operating these centralised control algorithms may cause the entire swarm to experience operational downtime. Attacks involving adversarial data injection into wireless packets are beyond the scope of this work.

Fortunately, several studies have shown that GNNs exhibit transferability across diverse graphs [9]. This property has been analysed using various approaches, including the sampling method [9], the stability approach [38], and the graphon method [39]. However, these studies have thus far been limited to open-loop systems. Integrating GNNs into closed-loop systems poses new theoretical challenges, particularly in the presence of adversarial attacks (i.e. dynamic changes in the network topology during inference). To the best of our knowledge, the resilience of GNNs in closed-loop systems remains an open question. In addition to certifying the transferability of graph neural networks, adversarial training has also been used to improve the resilience of GNN algorithms [40], [41]. Adversarially perturbed graphs are injected into the training dataset to enhance GNN performance during inference [42]. Therefore, this work focuses on adopting an adversarial training approach to optimise the parameters of GNNs and empirically evaluate their resilience under the temporary or permanent communication link failures.

B. Contributions and Paper Organisation

In this study, we propose leveraging graph neural networks to achieve resilient clock synchronisation among UAVs. The heat kernel retains low-frequency graph signals while attenuates high-frequency graph signals, which is consistent with the aim of time synchronisation. We integrate the heat kernel into the graph neural networks to improve clock synchronisation performance in swarms. Then, we introduce the distributed GNN architecture, which can avoid significant communication overhead, compared to the existing decentralised GNNs that rely on fully-connected networks. Through the adversarial training scheme, our GNN-based control policies achieve similar synchronisation performance without requiring re-training when scaled to large swarms, as compared to the centralised controller using fully-connected wireless networks. Furthermore, once trained, the proposed GNN-based control policies are also resilient to varying wireless networks, including the temporary or permanent communication link failures resulting from jamming attacks, and can maintain synchronisation even when the swarm is split into two disconnected parts.

The rest of this paper is organised as follows: Section 2 presents the preliminaries and problem description. Next, Section 3 introduces the distributed graph neural network architecture using the heat kernel, which is trained using the adversarial imitation learning scheme. Then, the simulation results are shown in Section 4. Finally, Section 5 concludes this work.

II. Preliminaries and Problem Description

This section starts by presenting the widely used clock model and the centralised networked controller based on a

[2]High and low frequencies in graph signals refer to variations between nodes in a graph, rather than to temporal frequencies [e.g. f_0 and $f[t]$ in Equ. (1), with units of Hz], as in traditional signal processing. Low-frequency graph signals represent that all nodes possess similar values, and show only minor variations among adjacent nodes. Further details on high- and low-frequency graph signals are provided in Section 3.

TABLE II
MAIN NOTATION TABLE.

Symbol	Description
$\theta[t]$, $\gamma[t]$	Clock offset and skew at the time instant t.
$\omega_\theta[t]$, $\omega_\gamma[t]$	Clock offset and skew noises at t.
$u_\theta[t]$, $u_\gamma[t]$	Offset and skew control inputs at t.
τ	Clock update interval.
$\mathbf{x}[t]$	Networked clock state at t, including all clock states $x[t] = [\theta[t],\ \gamma[t]]^\top$.
$\mathbf{u}[t]$	Networked control input at t, consisting of all control inputs $u[t] = [u_\theta[t],\ u_\gamma[t]]^\top$.
$\boldsymbol{\omega}[t]$	Networked clock noise at t, containing all noise components $\omega[t] = [\omega_\theta[t],\ \omega_\gamma[t]]$.
$\mathbf{L}_{\mathsf{full}}$	Swarm Laplacian matrix for a fully-connected wireless network.
$\mathcal{F}^*$	The centralised controller.
$\mathbf{L}[t]$	Time-varying Laplacian matrix at t, induced by mobility and wireless link failures.
$\mathbf{H}_\ell$	Feature matrix with G_ℓ hidden features at the ℓ-th GNN layer.
$\mathbf{W}$	A set of learnable weight matrices $\mathbf{W}_{\ell,k}$ for the k-th polynomial term at the ℓ-th GNN layer.
ρ	Heat kernel hyperparameter.
L	Number of GNN layers.
$c[t]$	Cost for GNN evaluation and testing at t.
$\Delta[t]$	Synchronisation precision at the time instant t.

fully-connected communication network, which is considered the expert control policy in imitation learning. We then briefly introduce decentralised GNN-based control policies and describe the problem addressed in this work.

A. Basics of Conventional Time Synchronisation

In the wireless network of an aerial swarm with N UAVs, if two UAVs are within the communication radius R of each other, they can establish a bidirected communication link to exchange wireless packets. The weight a_{pq}, which is an element of $\mathbf{A}$, is equal to 1 (i.e. $a_{pq} = a_{qp} = 1$); otherwise, it is zero. $\mathbf{A}$ indicates the adjacency matrix of the swarm's wireless network, and $\mathbf{D}$ is the diagonal degree matrix of the communication network. The network Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The eigenvalues and eigenvectors of the Laplacian matrix $\mathbf{L}$ are denoted by $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_N)$ and $\mathbf{U} = (\boldsymbol{u}_1,\ \boldsymbol{u}_2,\ \cdots,\ \boldsymbol{u}_N)$, respectively, and we have $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. The i-th column of $\mathbf{U}$ (i.e. $\boldsymbol{u}_i$) corresponds to the frequency component associated with eigenvalue λ_i. A small eigenvalue means that the corresponding eigenvector is a low-frequency graph signal, while a large eigenvalue represents a high-frequency graph signal. The non-negative eigenvalues are typically ordered as $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$. The orthogonality of $\mathbf{U} = (\boldsymbol{u}_1,\ \boldsymbol{u}_2,\ \ldots,\ \boldsymbol{u}_N)$ implies that $\boldsymbol{u}_i^\top \boldsymbol{u}_j = 0$ when $i \neq j$, and $\boldsymbol{u}_i^\top \boldsymbol{u}_i \neq 0$. The *graph Fourier transform* of a graph signal $\mathbf{x} \in \mathbb{R}^N$ is $\tilde{\mathbf{x}} = \mathbf{U}^\top \mathbf{x}$, and its *inverse graph Fourier transform* is given by $\mathbf{x} = \mathbf{U}\tilde{\mathbf{x}}$. Here, $\tilde{\mathbf{x}}_\lambda = \mathbf{U}_{:\lambda}^\top \mathbf{z}$ represents the frequency component of $\mathbf{x}$ at the frequency λ, and $\mathbf{U}_{:\lambda}$ is the corresponding eigenvector. The main symbols are summarised in Table II.

For an embedded system, the clock module is composed of a crystal oscillator and a chain of counters [12]. At the t-th *reference time* instant, provided by an ideal clock with nominal frequency f_0, the local time of a non-ideal clock module operating at frequency $f[t]$ is commonly denoted by $C[t]$. The behaviour of a free-running non-ideal clock is typically modelled as a double integrator:

$$\begin{cases} \theta[t+1] = \theta[t] + \tau\gamma[t] + \omega_\theta[t] \\ \gamma[t+1] = \gamma[t] + \omega_\gamma[t] \end{cases} \tag{1}$$

where $\theta[t]$ and $\gamma[t]$ represent the clock offset and skew, respectively, at the t-th time instant. The clock offset is defined as $\theta[t] = C[t] - C_0[t]$, where $C_0[t]$ is the ideal clock time. The clock skew is given by $\gamma[t] = (f[t] - f_0)/f_0$. τ is the clock update interval (or step size). The offset and skew noises $\omega_\theta[t]$ and $\omega_\gamma[t]$ are modelled as uncorrelated white Gaussian processes with variances $\tau\sigma_\theta^2$ and $\tau\sigma_\gamma^2$, respectively [43].

If no synchronisation solution is employed in the swarm, each UAV possesses a non-ideal clock indicating a different time $C[t]$, owing to the presence of $\theta[0]$, $\gamma[0]$, $\omega_\theta[t]$ and $\omega_\gamma[t]$. However, through wireless exchange of clock information among UAVs, the clock offset and skew can be measured, thereby obtaining the clock offset adjustment value $u_\theta[t]$ and the skew adjustment value $u_\gamma[t]$. The aerial swarm achieves time synchronisation by applying $u_\theta[t]$ and $u_\gamma[t]$ to each UAV clock. Similar to [7], we modify Equ. (1) to model the synchronisation of networked clocks:

$$\mathbf{x}[t+1] = \mathbf{x}[t]\mathcal{A} + \mathbf{u}[t] + \boldsymbol{\omega}[t], \tag{2}$$

where $\mathbf{x}[t] = \left[x_1^\top[t],\ x_2^\top[t],\ \ldots,\ x_N^\top[t]\right]^\top \in \mathbb{R}^{N\times 2}$ is the networked clock state (which also represents the graph signals), and the local clock state $x[t]$ is given by $x[t] = [\theta[t],\ \gamma[t]]^\top$. Similarly, $\mathbf{u}[t] = \left[u_1^\top[t],\ u_2^\top[t],\ \ldots,\ u_N^\top[t]\right]^\top \in \mathbb{R}^{N\times 2}$ denotes the networked control input, and $u[t]$ is defined as $u[t] = [u_\theta[t],\ u_\gamma[t]]^\top$, with values determined by different control strategies. The networked clock noise is given by $\boldsymbol{\omega}[t] = \left[\omega_1^\top[t],\ \omega_2^\top[t],\ \ldots,\ \omega_N^\top[t]\right]^\top \in \mathbb{R}^{N\times 2}$, and $\omega[t]$ is equal to $\omega[t] = [\omega_\theta[t],\ \omega_\gamma[t]]$. The system matrix $\mathcal{A}$ is given by

$$\mathcal{A} = \begin{bmatrix} 1 & 0 \\ \tau & 1 \end{bmatrix}.$$

Until now, many control strategies have been proposed for calculating $\mathbf{u}[t]$. To achieve time synchronisation, we adopt a centralised networked controller based on a fully-connected communication network for imitation learning:

$$\begin{aligned} \mathbf{u}[t] &= \mathcal{F}^*(\mathbf{x}[t] \mid \mathbf{L}_{\mathsf{full}}) \\ &= \mathbf{L}_{\mathsf{full}}\mathbf{x}[t]\mathbf{K} \end{aligned} \tag{3}$$

where $\mathbf{L}_{\mathsf{full}}$ denotes the Laplacian matrix of the swarm fully-connected wireless network, $\mathbf{K}$ is the networked control gain. The networked clock state $\mathbf{x}[t]$ is used to calculate the networked control input $\mathbf{u}[t]$. In the following, the above centralised networked controller $\mathcal{F}^*$ in Equ. (3) is referred to as the expert control policy for imitation learning.

By combining Equs. (2) and (3), the closed-loop networked clock synchronisation system under the centralised networked controller is obtained as follows:

$$\begin{cases} \mathbf{x}[t+1] = \mathbf{x}[t]\mathcal{A} + \mathbf{u}[t] + \boldsymbol{\omega}[t] \\ \mathbf{u}[t] = \mathbf{L}_{\mathsf{full}}\mathbf{x}[t]\mathbf{K} \end{cases} \tag{4}$$

Utilising the fully-connected wireless network leads to significant communication overhead, particularly for a swarm with a large number of UAVs. Thus, a fully-connected wireless network is uncommon in practice. In addition, the network topology may vary over time due to the high mobility of UAVs and the presence of the communication link failures. The swarm network Laplacian matrix $\mathbf{L}[t]$ at the t-th time instant is given by:

$$\mathbf{L}[t] = \mathbf{L}_{\mathsf{init}}[t] + \Delta\mathbf{L}_{\mathsf{var}}[t] + \Delta\mathbf{L}_{\mathsf{jam}}[t] \tag{5}$$

where $\mathbf{L}_{\mathsf{init}}[t]$ is the initial Laplacian matrix, $\Delta\mathbf{L}_{\mathsf{var}}[t]$ implies the topological perturbations caused by frequent changes in the highly mobile wireless network, and $\Delta\mathbf{L}_{\mathsf{jam}}[t]$ represents the temporary or permanent communication failures resulting from jamming attacks. If the communication link between two UAVs is disrupted by a jamming attack, the corresponding off-diagonal elements in the Laplacian and adjacency matrices become zero (i.e. $a_{pq} = a_{qp} = 0$).

By incorporating the time-varying wireless network topology $\mathbf{L}[t]$, Equ. (4) is modified as follows:

$$\begin{cases} \mathbf{x}[t+1] = \mathbf{x}[t]\mathcal{A} + \mathbf{u}[t] + \boldsymbol{\omega}[t] \\ \quad \mathbf{u}[t] = \mathbf{L}[t]\mathbf{x}[t]\mathbf{K}[t] \end{cases} \tag{6}$$

Typically, designing the networked control gain requires consideration of the network topology. The value of $\mathbf{K}[t]$ depends on the Laplacian matrix, and any change in the matrix necessitates a re-design of $\mathbf{K}[t]$ [7].

To summarise, conventional control methods are usually designed in a centralised manner based on a known network topology [e.g. $\mathbf{L}[0]$ in Equ. (6)]. The pre-determined control gain $\mathbf{K}[0]$ may not be applicable to an aerial swarm with a time-varying communication network, and a re-design of $\mathbf{K}[t]$ is required at the time instant t. Moreover, an attack on the central UAV operating these centralised control algorithms may lead to the downtime of the entire swarm system. Our interest lies in using imitation learning to let the GNN-based control policies mimic the centralised controller in Equ. (3). The trained GNN models are resilient to topological changes in the swarm communication network, such as those caused by the temporary or permanent communication link failures.

B. Problem Description

Thanks to the inherently transferable and decentralised nature of graph neural networks, GNN-based control policies may overcome the above issues associated with the traditional control methods. However, the decentralised GNN-based control policies that rely on fully-connected wireless networks also result in significant communication overhead in a swarm with a large number of UAVs. In addition, the GNNs composed of graph matrix polynomials tend to amplify high-frequency graph signals while suppressing low-frequency components. This behaviour conflicts with the purpose of synchronisation, which involves keeping only low-frequency graph signals.

Our goal is twofold. First, we propose distributed GNN-based control policies that integrate the heat kernel and imitate the centralised networked controller $\mathcal{F}^*$ in Equ. (3). The proposed distributed GNN architecture involves communication only among one-hop neighbouring UAVs. The integration of the heat kernel into the GNNs explicitly attenuates high-frequency graph signals. Consequently, the trained GNN models achieve similar synchronisation performance without requiring re-training when applied to a swarm with large-scale or changing wireless network due to the high mobility, as compared to the centralised controller using a fully-connected communication network. Second, our GNN models trained through adversarial imitation learning are resilient to a varying wireless network, particularly in the presence of the temporary or permanent link failures.

Therefore, the proposed GNN-based control policies for time synchronisation ensure that all UAVs maintain identical offset and skew values, even in swarms with large-scale structures or communication networks subject to topological changes due to the high mobility or the communication link failures. That is,

$$\lim_{t\to\infty} \mathbf{x}^\top[t]\mathbf{L}[t]\mathbf{x}[t] = \mathbf{0}_{2\times 2} \tag{7}$$

where $\mathbf{0}_{2\times 2}$ is a 2×2 diagonal matrix with all diagonal entries equal to zero.

Remark 1. The random fluctuations of the clock frequency $f[t]$ in Equ. (1) are dominated by random-walk Frequency Modulation (FM) noise and white FM noise [43]. It is usually sufficient to consider these two types of noise to represent the dynamics of the clock frequency [44]. For example, the clock models in [43] and [44], and the references therein, all assume that the clock noise is Gaussian. The finite-order state model fails to model flicker FM noise, and the double-integrator state-space model in Equ. (1) also cannot include this noise [43], [45].

The impacts of Phase Modulation (PM) noise are short-term, usually affecting timescales shorter than the typical time synchronisation cycle of $1\,\mathrm{s}$ [43], which corresponds to the order of magnitude commonly adopted in wireless communications, such as in the IEEE 1588 synchronisation protocol. Accordingly, it is reasonable to assume that such synchronisation schemes have a negligible effect on the performance of the local clock with respect to both types of phase noise. Therefore, their contributions are not considered in our clock model.

Remark 2. The derivation of the networked control gains in Equs. (4) and (6) is beyond the scope of our study. We aim to use GNNs to imitate the centralised networked controller. Readers interested in the design and derivation of $\mathbf{K}[t]$ are referred to our prior work [7], where we design a networked controller guaranteeing that the achieved time synchronisation performance is robust to the networked clock noise $\boldsymbol{\omega}[t]$.

III. Distributed Graph Neural Networks

In this section, we first analyse the impact of the `Tanh` activation function in GNNs from a spectral perspective, and integrate the heat kernel into graph neural networks. Then, we introduce the distributed graph neural network architecture. Finally, we leverage adversarial imitation learning to train the

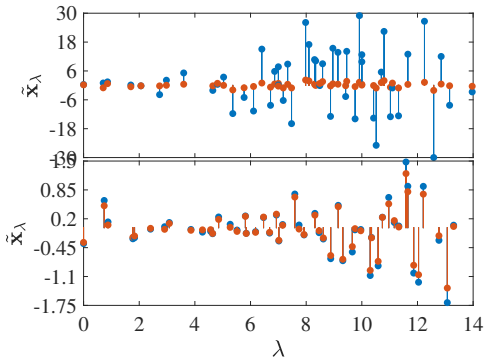

Fig. 1. Frequency components of graph signals before and after applying the Tanh function at $t = 2$ (upper) and $t = 900$ (lower). Blue represents the graph signal components before applying Tanh, while orange corresponds to the components after applying Tanh.

proposed GNNs, resulting in the GNN models that are resilient to varying swarm networks caused by the communication link failures.

A. Graph Neural Networks with the Heat Kernel

So far, GNNs have been extensively studied from both the spectral and spatial perspectives. The inference process of GNNs is usually divided into two phases. The aggregation and combination phase consists of L layers, with each layer represented as a K-th order polynomial of the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N\times N}$, yielding:

$$\mathbf{H}_{\ell+1} = \boldsymbol{\sigma}\left(\sum_{k=0}^{K} \mathbf{L}^k \mathbf{H}_\ell \mathbf{W}_{\ell,k}\right) \tag{8}$$

where $\mathbf{H}_\ell = \left[\mathbf{h}_\ell^1,\ \mathbf{h}_\ell^2,\ \ldots,\ \mathbf{h}_\ell^{G_\ell}\right] \in \mathbb{R}^{N\times G_\ell}$ represents the feature matrix at the ℓ-th layer, and G_ℓ is the number of (hidden) features at this layer. $\mathbf{H}_\ell$ contains G_ℓ graph signals [46], with each column $\mathbf{h}_\ell^g$ representing a graph signal corresponding to one feature at the ℓ-th layer. $\mathbf{W}_{\ell,k} \in \mathbb{R}^{G_\ell \times G_{\ell+1}}$ denotes the learnable weight matrix corresponding to the k-th polynomial term at the ℓ-th layer, and $\boldsymbol{\sigma}$ is the element-wise activation function.

In [10], [31], [32], [47], $\boldsymbol{\sigma}$ in Equ. (8) is the tangent activation function (i.e. Tanh). We analyse the role of Tanh in GNNs from the spectral perspective. Given the element-wise Tanh operation applied to the graph signals $\mathbf{x}[t]$ in the spatial domain, we investigate its equivalent effects, denoted as Tanh′, on $\tilde{\mathbf{x}}[t]$ in the spectral domain, following

$$\texttt{Tanh}'\left(\tilde{\mathbf{x}}[t]\right) = \mathbf{U}^\top \texttt{Tanh}\left(\mathbf{U}\tilde{\mathbf{x}}[t]\right) \tag{9}$$

This implies that different frequency components are first transformed via $\mathbf{U}$, then attenuated through the element-wise Tanh function, and finally redistributed to each frequency via $\mathbf{U}^\top$. Fig. 1 shows that Tanh typically suppresses the frequency components, especially those corresponding to large frequency values of λ. This is particularly true during convergence (i.e. $t = 2$). However, suppressing the frequency components may increase the convergence time. We conclude that the Tanh function is not critical, which is also consistent with the results in [18]. Therefore, no activation function is used in our work.

In the following, we remove the Tanh activation function from GNNs, and analyse Equ. (8) in the spectral domain:

$$\begin{aligned}
\mathbf{H}_{\ell+1} &= \sum_{k=0}^{K} \mathbf{L}^k \mathbf{H}_\ell \mathbf{W}_{\ell,k} \\
&= \mathbf{H}_\ell \mathbf{W}_{\ell,0} \\
&+ \left(\lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^\top + \cdots + \lambda_N \boldsymbol{u}_N \boldsymbol{u}_N^\top\right) \mathbf{H}_\ell \mathbf{W}_{\ell,1} \\
&+ \cdots + \left(\lambda_1^K \boldsymbol{u}_1 \boldsymbol{u}_1^\top + \cdots + \lambda_N^K \boldsymbol{u}_N \boldsymbol{u}_N^\top\right) \mathbf{H}_\ell \mathbf{W}_{\ell,K} \\
&= \mathbf{H}_\ell \mathbf{W}_{\ell,0} \\
&+ \boldsymbol{u}_1 \boldsymbol{u}_1^\top \mathbf{H}_\ell \left(\lambda_1 \mathbf{W}_{\ell,1} + \cdots + \lambda_1^K \mathbf{W}_{\ell,K}\right) \\
&+ \cdots + \boldsymbol{u}_N \boldsymbol{u}_N^\top \mathbf{H}_\ell \left(\lambda_N \mathbf{W}_{\ell,1} + \cdots + \lambda_N^K \mathbf{W}_{\ell,K}\right) \\
&= \mathbf{U}\left(\sum_{k=0}^{K} \boldsymbol{\Lambda}^k \left(\mathbf{U}^\top \mathbf{H}_\ell\right) \mathbf{W}_{\ell,k}\right).
\end{aligned} \tag{10}$$

From Equ. (10), it can be seen that the graph Fourier transform is first utilised to convert the graph signals $\mathbf{H}_\ell \in \mathbb{R}^{N\times G_\ell}$ from the spatial domain to the spectral domain. Then, by applying the matrix polynomial $\sum_{k=0}^{K} \boldsymbol{\Lambda}^k(\cdot)\mathbf{W}_{\ell,k}$ and the inverse graph Fourier transform, we finally obtain the filtered graph signals $\mathbf{H}_{\ell+1} \in \mathbb{R}^{N\times G_{\ell+1}}$ in the spatial domain.

In Equ. (10), $\left\{\boldsymbol{u}_1\boldsymbol{u}_1^\top,\ \boldsymbol{u}_2\boldsymbol{u}_2^\top,\ \ldots,\ \boldsymbol{u}_N\boldsymbol{u}_N^\top\right\}$ is a set of basic filters [26], [27]. $\left(\lambda_i \mathbf{W}_{\ell,1} + \lambda_i^2 \mathbf{W}_{\ell,2} + \cdots + \lambda_i^K \mathbf{W}_{\ell,K}\right)$ represents the basic filter coefficient, which also indicates the magnitude of a graph signal that can pass through the filter $\boldsymbol{u}_i\boldsymbol{u}_i^\top$. Note that the message passing scheme (i.e. aggregation) in the spatial domain is essentially equivalent to filtering with N basic filters in the spectral domain. For each eigenvector $\boldsymbol{u}_i$ of the Laplacian matrix $\mathbf{L}$, its corresponding eigenvalue λ_i can characterise the smoothness of $\boldsymbol{u}_i$:

$$\lambda_i = \boldsymbol{u}_i^\top \mathbf{L} \boldsymbol{u}_i = \frac{1}{2} \sum_{p=1,q=1}^{N} a_{pq} \left(\boldsymbol{u}_i(p) - \boldsymbol{u}_i(q)\right)^2. \tag{11}$$

From Equ. (11), a small eigenvalue λ_i indicates that the corresponding graph signal $\boldsymbol{u}_i$ carries tiny variations across adjacent nodes, and is thus considered a low-frequency (i.e. smoother) graph signal. Conversely, a large eigenvalue for $\boldsymbol{u}_i$ suggests a high-frequency (i.e. less smooth) graph signal $\boldsymbol{u}_i$ [48], [49]. This implies that the high and low frequencies in this work refer to variations across the nodes of the graph, rather than to temporal frequencies [e.g. $f[t]$ in Equ. (1)].

If no time synchronisation algorithm is applied in aerial swarms, each UAV possesses distinct clock offset and skew values. The initial networked clock state $\mathbf{x}[0]$ on the graph corresponds to high-frequency graph signals. Once a synchronisation solution is employed and all clocks in the wireless communication network are synchronised, the UAVs share the same clock offset and skew values, resulting in $\mathbf{x}[\infty]$ becoming low-frequency graph signals. This implies that the purpose of time synchronisation is to evolve the high-frequency graph signals $\mathbf{x}[0]$ into the low-frequency graph signals $\mathbf{x}[\infty]$ at the steady synchronised state. However, the GNNs in [10], [31], [32], [47], which consist of several instances of Equ. (8), assign greater weight to high-frequency basic filters and

lesser weight to low-frequency ones. In other words, the GNNs in these studies amplify high-frequency signals while suppressing low-frequency graph signals. By contrast, the heat kernel smooths out the features of all nodes in a graph and retains only low-frequency graph signals, explicitly reducing the impact of high-frequency components [26], which is consistent with the aim of synchronisation. From [25], the heat kernel is defined as

$$f(\mathbf{\Lambda}) = e^{-\rho\mathbf{\Lambda}} = \begin{bmatrix} e^{-\rho\lambda_1} & 0 & \cdots & 0 \\ 0 & e^{-\rho\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-\rho\lambda_N} \end{bmatrix} \tag{12}$$

where $\rho > 0$ is the hyperparameter, and it controls the heat diffusion rate. We propose to use the heat kernel in Equ. (12) for graph neural networks, yielding:

$$\begin{aligned} \mathbf{H}_{\ell+1} &= \mathbf{U}\left(\sum_{k=0}^{K} f\left(\mathbf{\Lambda}\right)^k \left(\mathbf{U}^T\mathbf{H}_\ell\right)\mathbf{W}_{\ell,k}\right) \\ &= \mathbf{H}_\ell\mathbf{W}_{\ell,0} \\ &+ \left(e^{-\rho\lambda_1}\boldsymbol{u}_1{\boldsymbol{u}_1}^\top + \cdots + e^{-\rho\lambda_N}\boldsymbol{u}_N{\boldsymbol{u}_N}^\top\right)\mathbf{H}_\ell\mathbf{W}_{\ell,1} \\ &+ \cdots + \left(e^{-K\rho\lambda_1}\boldsymbol{u}_1{\boldsymbol{u}_1}^\top + \cdots\right. \\ &+ \left. e^{-K\rho\lambda_N}\boldsymbol{u}_N{\boldsymbol{u}_N}^\top\right)\mathbf{H}_\ell\mathbf{W}_{\ell,K} \\ &= \mathbf{H}_\ell\mathbf{W}_{\ell,0} + e^{-\rho\mathbf{L}}\mathbf{H}_\ell\mathbf{W}_{\ell,1} + \cdots + e^{-K\rho\mathbf{L}}\mathbf{H}_\ell\mathbf{W}_{\ell,K} \\ &= \sum_{k=0}^{K} e^{-k\rho\mathbf{L}}\mathbf{H}_\ell\mathbf{W}_{\ell,k} \end{aligned} \tag{13}$$

where $e^{-k\rho\lambda_i}$ is the weight assigned to the basic filter $\boldsymbol{u}_i{\boldsymbol{u}_i}^\top$, which decreases as λ_i increases. In Equ. (8), the eigenvalue λ_i corresponding to $\boldsymbol{u}_i$ represents the frequency of graph signals can pass through the filter $\boldsymbol{u}_i{\boldsymbol{u}_i}^T$. It can be seen that the proposed GNN integrating the heat kernel [i.e. Equ. (13)] can retain low-frequency signals while suppressing high-frequency graph signals; the above characteristic is missing in Equ. (8).

From the graph spectral perspective, the input to GNNs may contain both high- and low-frequency components. However, when using the heat kernel-based GNNs, the high-frequency components are exponentially attenuated due to the low-pass nature of the heat kernel [see Equ. (13)]. As a result, the output of the heat kernel-based GNNs mainly retains the low-frequency components of the input graph signals. It is important to note that low-frequency graph signals correspond to smooth, globally consistent variations across the nodes, rather than constant values. Thus, these smoothed GNN outputs ensure that all nodes (i.e. UAVs) adjust their local clocks in a coordinated manner, thereby gradually achieving time synchronisation in aerial swarms.

B. Distributed Graph Neural Networks for Synchronisation

Modifying the term $\mathbf{L}^k\mathbf{H}_\ell$ in Equ. (10) to the recursive form $\mathbf{L}\left(\mathbf{L}^{k-1}\mathbf{H}_\ell\right)$ allows GNNs to iteratively aggregate features from successive one-hop neighbours at each node until $k = K$. This means that $\mathbf{L}^k\mathbf{H}_\ell$ contains aggregated information from the k-hop neighbourhood of each node. Such a recursive form lets GNNs naturally exhibit the decentralised characteristic.

For the decentralised GNN approach, UAVs in a swarm are divided into multiple clusters. In each cluster, UAVs communicate with their corresponding cluster head. However, in [10], [50], a fully-connected wireless network is adopted in each cluster to enable communication among UAVs. As a result, this setup turns every UAV into a cluster head, which increases the communication overhead in the aerial swarm. Moreover, in [31], [32], [47], the GNN-based control policies in Equ. (8) are simulated in a centralised manner: for example, by pre-computing $\mathbf{L}^k\mathbf{H}_\ell$ prior to GNN inference [18]. Yet, simulating GNNs in these manners is not consistent with their decentralised nature [33]. Therefore, we set K in Equ. (10) to one (i.e. $K = 1$), following:

$$\mathbf{H}_{\ell+1} = \mathbf{H}_\ell\mathbf{W}_{\ell,0} + \mathbf{L}\mathbf{H}_\ell\mathbf{W}_{\ell,1} \tag{14}$$

Each layer in Equ. (14) aggregates only the hidden features of one-hop neighbours for each node. This implies that stacking L layers of Equ. (14) allows every node to aggregate the features of its L-hop neighbours [18], [29], [51], thereby realising distributed message passing and aggregation in GNNs. Likewise, through letting $K = 1$, Equ. (13) is modified to

$$\mathbf{H}_{\ell+1} = \mathbf{H}_\ell\mathbf{W}_{\ell,0} + e^{-\rho\mathbf{L}}\mathbf{H}_\ell\mathbf{W}_{\ell,1} \tag{15}$$

In the proposed GNN architecture, no activation function $\boldsymbol{\sigma}$ is included. We aim to apply GNNs to realise clock synchronisation in aerial swarms, which is formulated as a node-level regression task. Hence, there is no need for the MLP-based readout used in [31] and [47].

As shown in Fig. 2, each UAV in the swarm is modelled as a node in our GNN architecture, where the one-hop feature aggregation is equivalent to the one-hop packet exchange between neighbouring UAVs. Since both processes rely on local, one-hop interactions, the swarm's distributed communication network can be naturally represented by the GNN's message-passing scheme. By stacking multiple layers (e.g. L) in the form of Equ. (14), each node can aggregate information from its L-hop neighbourhood, without requiring direct communication through a fully connected network topology. Meanwhile, in the swarm communication network, each UAV exchanges packets only with its one-hop neighbours, and there is no reliance on any cluster head. This mutual alignment between the swarm communication pattern and the GNN architecture allows a distributed implementation, which in turn effectively reduces communication demands in the swarm. Moreover, there is no central or cluster-head UAV in our architecture, which prevents the swarm from experiencing downtime due to attacks on individual UAVs.

In Equ. (16), we build our proposed linear graph neural network, denoted by $\Phi_{\mathrm{l}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$, where $\mathbf{W} = \{(\mathbf{W}_{\ell,0},\ \mathbf{W}_{\ell,1}),\ \ell = 0,\ 1,\ \ldots,\ (L-1)\}$ is the set of learnable weights. The layer-wise computation of the proposed graph neural network at the t-th time instant is presented in Fig. 2. The GNN input is the networked clock state $\mathbf{x}[t]$, namely, $\mathbf{H}_0 = \mathbf{x}[t]$. Each layer performs a one-hop feature aggregation to compute the next hidden features. This process is repeated L times, resulting in the output $\mathbf{H}_L = \mathbf{u}[t]$, which contains the control input $u[t]$ for each UAV. The GNN is parameterised by the weights $\mathbf{W}$, which are optimised offline

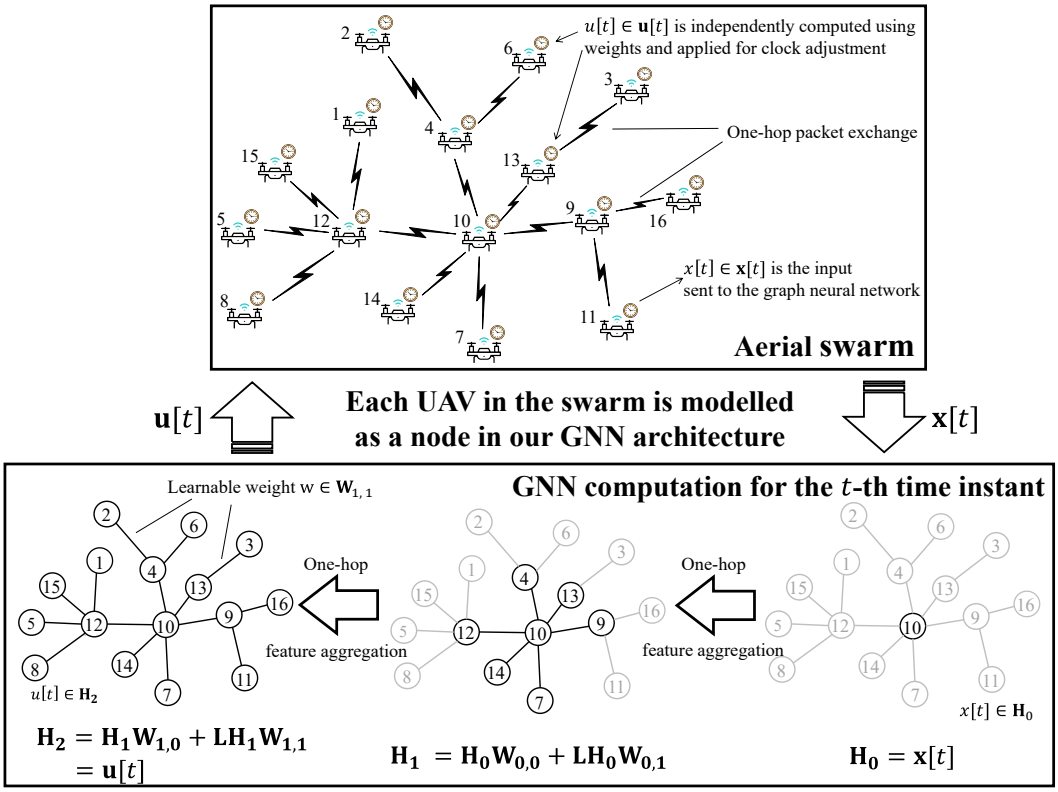

Fig. 2. Time synchronisation in an aerial swarm using the distributed graph neural network architecture. Each UAV's clock is represented as a node in the GNN, while communications between neighbouring UAVs are modelled via learnable weights. Stacking multiple GNN layers allows multi-hop information aggregation, which is similar to multi-hop wireless communication within the swarm.

$$\begin{cases} \mathbf{x}[t+1] = \mathbf{x}[t]\mathcal{A} + \mathbf{u}[t] + \boldsymbol{\omega}[t], \\ \mathbf{u}[t] = \sum_{k=0}^{1} \mathbf{L}[t]^k \left(\cdots \left(\sum_{k=0}^{1} \mathbf{L}[t]^k \mathbf{x}[t] \mathbf{W}_{0,k} \right) \cdots \right) \mathbf{W}_{L-1,k} \end{cases} \tag{16}$$

$$\begin{cases} \mathbf{x}[t+1] = \mathbf{x}[t]\mathcal{A} + \mathbf{u}[t] + \boldsymbol{\omega}[t], \\ \mathbf{u}[t] = \sum_{k=0}^{1} e^{-k\rho\mathbf{L}[t]} \left(\cdots \left(\sum_{k=0}^{1} e^{-k\rho\mathbf{L}[t]} \mathbf{x}[t] \mathbf{W}_{0,k} \right) \cdots \right) \mathbf{W}_{L-1,k} \end{cases} \tag{17}$$

by minimising a training loss [see Equ. 19] to imitate the expert controller $\mathcal{F}^*$. During inference, each UAV only needs to store the relevant weights for its local neighbourhood and can independently compute its control input $u[t]$, thereby enabling distributed synchronisation across the swarm.

Similarly, $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ is used to denote our GNN based on the heat kernel [see Equs. (15) and (17)]. The closed-loop networked synchronisation systems under these GNN-based control policies are described in Equs. (16) and (17). For clarity, we adopt $\Phi\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ to represent the proposed distributed GNNs, namely, $\Phi_{\mathsf{l}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ and $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$, in the following.

C. Adversarial Imitation Learning

In addition to introducing the distributed graph neural network architecture, we also investigate the resilience of GNNs to varying wireless networks caused by the high mobility or the communication link failures. The graph neural networks are trained using both imitation learning and adversarial training. Imitation learning allows the model to learn a policy that mimics the behaviour of the centralised controller from demonstrations. Adversarial training optimises the GNN parameters by using perturbed input graphs, thereby enhancing the model's resilience to changes in the network topology. Typically, this training strategy is formulated as a bi-level min-max optimisation problem, consisting of two iterative processes: (i) generating perturbations $\delta_{\mathbf{L}}$ to maximise $\|\Phi\left(\mathbf{x}[t],\ (\mathbf{L}+\delta_{\mathbf{L}})\,;\ \mathbf{W}\right) - \mathcal{F}^*\left(\mathbf{x}[t] \mid \mathbf{L}_{\mathsf{full}}\right)\|$, and (ii) updating model parameters $\mathbf{W}$ to minimise the same objective.

In this study, we leverage imitation learning to train GNNs $\Phi\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ from a dataset composed of input-output pairs from Equ. (4). Here, the input refers to the value sent to the centralised controller $\mathcal{F}^*$ [i.e. $\mathbf{x}[t]$ in Equ. (4)], while the output corresponds to the value computed by $\mathcal{F}^*$ using a fully-connected wireless communication network [i.e. $\mathbf{u}[t]$ in Equ.

(4)]. Instead of adversarially injecting topology perturbations to solve the first maximisation problem, we use the time-varying Laplacian matrices $\{\mathbf{L}[t]\}$ from Equ. (5) to optimise the model parameters by minimising $\|\Phi(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}) - \mathcal{F}^*(\mathbf{x}[t] \mid \mathbf{L}_{\mathsf{full}})\|$.

To do so, we minimise the expected norm of the difference between the GNN outputs $\Phi(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$ and the centralised controller outputs $\mathcal{F}^*(\mathbf{x}[t] \mid \mathbf{L}_{\mathsf{full}})$ over the probability distribution of $\mathbf{x}[t]$ under the centralised controller, yielding:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \mathbb{E}\left[\left\|\Phi(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}) - \mathcal{F}^*(\mathbf{x}[t] \mid \mathbf{L}_{\mathsf{full}})\right\|\right] \tag{18}$$

where $\|\cdot\|$ is the norm of graph signals. For example, consider graph signals $\mathbf{H} \in \mathbb{R}^{N\times G}$ with G features, the norm of $\mathbf{H}$ equals $\|\mathbf{H}\| = \sum_{g=1}^{G} \|\mathbf{h}^g\|_2$. The probability distribution of $\mathbf{x}[t]$ under the centralised controller $\mathcal{F}^*$ is usually unknown. To address this issue, we collect M samples of the pairs $(\mathbf{x}_m[t], \mathbf{u}_m[t])$ from Equ. (4) under the centralised controller $\mathcal{F}^*$. By applying the law of large numbers [39], it is reasonable to approximate Equ. (18) using the following form:

$$\widehat{\mathbf{W}}^* = \arg\min_{\mathbf{W}} \frac{1}{M}\sum_{m=1}^{M}\left[\left\|\Phi(\mathbf{x}_m[t],\ \mathbf{L}[t];\ \mathbf{W}) - \mathcal{F}^*(\mathbf{x}_m[t] \mid \mathbf{L}_{\mathsf{full}})\right\|\right] \tag{19}$$

where $\widehat{\mathbf{W}}^*$ in Equ. (18) is close to $\mathbf{W}^*$ in Equ. (19) when M is sufficiently large. $\mathbf{L}[t]$ represents the changing wireless networks resulting from the high mobility of UAVs, which are not affected by the communication link failures during training. The validation and test graphs (i.e. swarm network Laplacian matrices) are excluded during training.

For the non-convex optimisation problem in Equ. (19), the Stochastic Gradient Descent (SGD) algorithm [26], [47] is commonly used to find the learnable parameters $\mathbf{W}$, letting the trained model generalise well to the test set [31]. This allows the trained GNN model to mimic the behaviour of the centralised controller $\mathcal{F}^*$. Even though $\mathbf{W}$ and $\mathbf{K}$ differ in form, they yield similar input-output pairs, thereby achieving comparable control performance. The networked control gain $\mathbf{K}$ of the centralised controller is designed to guarantee the stability of the closed-loop synchronisation system in Equ. (4); therefore, the GNN-based control policy is expected to inherit this stability property [52].

The GNN models trained via the adversarial imitation learning scheme, by minimising Equ. (19) under changing networks caused by the high UAV mobility, demonstrate resilience to the wireless link failures induced by jamming attacks, as shown in the simulation results. However, the integration of GNNs into the closed-loop systems introduces new challenges for theoretical analysis, particularly in terms of resilience under time-varying network topologies. To the best of our knowledge, the resilience of GNNs in such systems remains an open question. While we rely on empirical evidence to support the resilience of our distributed GNN-based control policies, the certification of their resilience in the closed-loop systems with time-varying network topologies will be addressed in future work.

During validation and testing, in order to evaluate the time synchronisation performance of a given controller in a swarm, we use the following metric:

$$\begin{aligned} c[t] = \frac{1}{Q}\sum_{\alpha\in\mathcal{U}[t]}\left(\theta_\alpha[t] - \frac{1}{Q}\sum_{\beta\in\mathcal{U}[t]}\theta_\beta[t]\right)^2 \\ + \frac{1}{Q}\sum_{\alpha\in\mathcal{U}[t]}\left(\tau\gamma_\alpha[t] - \frac{1}{Q}\sum_{\beta\in\mathcal{U}[t]}\tau\gamma_\beta[t]\right)^2 \end{aligned} \tag{20}$$

where $\mathcal{U}[t]$ denotes the set of unattacked UAVs at the time step t, $Q = |\mathcal{U}[t]|$ is the number of such UAVs. $\theta_\alpha[t]$ and $\gamma_\alpha[t]$, respectively, are the clock offset and skew of the α-th UAV's clock [see Equ. (1)]. The cost in Equ. (20) measures the deviation of each unattacked UAV's clock offset and skew from the average offset and skew in the swarm. The accumulated cost $\sum_{t=0}^{T} c[t]$ reflects the convergence speed of clock offsets and skews in the swarm, and also reports how good the synchronisation objective is achieved in the end. Note that Equ. (20) is equivalent to Equ. (7), as both quantify the deviation from the average value among unattacked UAVs over time. The proposed graph neural networks $\Phi(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$ are trained offline, with only inference performed online.

IV. Simulations

Here, we evaluate the performance of our distributed graph neural networks, which are implemented in PyTorch 1.11.0[3], for time synchronisation in aerial swarms. First, we analyse the effects of hyperparameters on time synchronisation performance. Then, we compare the proposed distributed GNNs with several approaches. Finally, we empirically demonstrate the resilience of the proposed GNN-based control policies to changes in the swarm wireless network caused by the communication link failures.

In the simulations, the aerial swarm contains 50 UAVs, each represented by the kinematic model in [11]. The flocking and collision avoidance algorithm, along with the swarm kinematic parameters, is adopted from [31]. The swarm flocking duration and clock update period (i.e. step size) are set to $10\,\mathrm{s}$ and $10\,\mathrm{ms}$, respectively. If two UAVs are within the wireless communication radius R (where $R = 2\,\mathrm{m}$) [31], they can establish a communication link, thus forming a bi-directional swarm wireless network. We normalise the Laplacian matrix $\mathbf{L}[t]$ by its maximum eigenvalue. In the swarm, each UAV is equipped with a clock modelled by Equ. (1). The initial clock offset and skew are set to $100\,\mu\mathrm{s}$ and 0 parts per million (ppm), respectively, and are further randomised using normal distributions with zero means and standard deviations of $50\,\mu\mathrm{s}$ and 50 ppm, respectively [8]. Clock offset and skew noises are not simulated in this work.

We adopt the above configurations to generate the dataset, which consists of 400 clock synchronisation trajectories for training, 20 time synchronisation trajectories for validation, and 20 synchronisation trajectories for testing. We train the GNNs for 30 epochs using the Adam optimiser with a learning

[3]https://github.com/zongyan/GNNSync

TABLE III
AVERAGE AND STANDARD DEVIATION OF $\sum_{t=0}^{T} c[t]$ FOR DIFFERENT HYPERPARAMETERS OF $\Phi_{\mathsf{h}}(\mathbf{H}; \mathbf{L}, \mathbf{W})$.

G \ L	2	3	4
16	3.96 ± 1.44	12.33 ± 3.96	568.73 ± 1061.95
32	1.93 ± 0.32	2.95 ± 0.68	210.99 ± 296.71
64	$\mathbf{1.14 \pm 0.21}$	1.36 ± 0.31	20.14 ± 8.50

TABLE IV
MEAN AND STANDARD DEVIATION OF $\sum_{t=0}^{T} c[t]$ FOR DIFFERENT HYPERPARAMETERS OF $\Phi_{\mathsf{l}}(\mathbf{H}; \mathbf{L}, \mathbf{W})$.

G \ L	2	3	4
16	4.14 ± 3.15	6.87 ± 5.02	13.60 ± 13.64
32	1.69 ± 0.57	1.22 ± 0.17	1.17 ± 0.15
64	1.11 ± 0.18	1.00 ± 0.16	$\mathbf{0.99 \pm 0.15}$

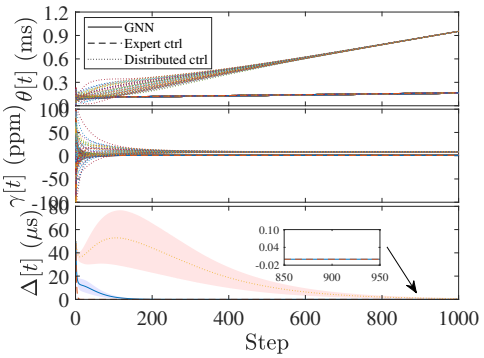

Fig. 3. Evolution of the clock offset, skew, and precision under the proposed distributed GNN-based control policy $\Phi_{\mathsf{h}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ (solid line), the centralised controller using a fully-connected wireless network $\mathcal{F}^*$ (dashed line) and the conventional distributed controller (dotted line).

rate of 0.0005. The loss function for imitation learning is the mean squared error between the GNN output and the centralised controller output [which is the same as in Equ. (19)]. The batch size is set to 20, and every 5 training steps, $\sum_{t=0}^{T} c[t]$ is calculated on the validation set. We utilise the GNN models that achieve the lowest value of $\sum_{t=0}^{T} c[t]$ on the validation set for testing. We report the mean and standard deviation of the accumulated cost $\sum_{t=0}^{T} c[t]$ across 10 randomly generated dataset realisations.

In addition, there is no leader UAV in the swarm, and all the algorithms aim to synchronise the UAV clocks to achieve consensus during the simulations. We define the following precision metric, $\Delta[t]$, to evaluate the synchronisation performance [53]:

$$\Delta[t] = \sqrt{\frac{1}{Q} \sum_{\alpha \in \mathcal{U}[t]} \left(\theta_\alpha[t] - \frac{1}{Q} \sum_{\beta \in \mathcal{U}[t]} \theta_\beta[t] \right)^2}. \tag{21}$$

A. Hyperparameter Selection

The heat kernel hyperparameter ρ is set to $\rho = 1$. We test different values of the hidden feature dimension $G_\ell \in \{16,\ 32,\ 64\}$ and the number of layers $L \in \{2,\ 3,\ 4\}$ for the distributed GNNs with and without the heat kernel. Tables III and IV summarise the results. The accumulated cost of the GNNs using the heat kernel decays as the number of layers increases, whereas the linear GNNs without the heat kernel exhibit the opposite behaviour. This is due to the non-linearities introduced by the heat kernel [see Equs. (14) and (15)]. We also find that utilising more hidden features can enhance GNN performance. In the following simulations, we select $G_\ell = 64$ and $L = 2$ for the GNNs $\Phi_{\mathsf{h}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ with the heat kernel, whose layer configuration L coincides with that of the GNNs in [26]. The value of G_ℓ and L for the linear GNN $\Phi_{\mathsf{l}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ without the heat kernel are set to $G_\ell = 64$ and $L = 4$, respectively.

B. Distributed GNN-based Control for Time Synchronisation

In order to evaluate the performance of the proposed distributed GNN-based control policies, we compare them with several control approaches: (i) the centralised controller, (ii) a conventional distributed controller, and (iii) state-of-the-art GNN-based control policies. Moreover, we investigate the scalability of our proposed distributed graph neural networks across different swarm sizes.

1) Performance Comparison between the Proposed GNN-based and Existing Controllers: First, we compare the distributed GNN-based control policies with the centralised controller $\mathcal{F}^*$, which uses a fully-connected communication network, and the distributed controller [8]. Fig. 3 presents the evolution of the clock offset, skew, and synchronisation precision under our distributed GNN-based control policy $\Phi_{\mathsf{h}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ and two traditional controllers.

We train the proposed distributed graph neural networks to learn a control policy that mimics the behaviour of the centralised controller $\mathcal{F}^*$ using a fully-connected communication network. In terms of the offset and skew, both $\Phi_{\mathsf{h}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ and $\mathcal{F}^*$ achieve similar performance. As illustrated in Fig. 3, the mean precision $\Delta[t]$ in the synchronised state (i.e. from step 900 to 1000, corresponding to 9 s to 10 s) across 10 randomly generated realisations is comparable under both $\Phi_{\mathsf{h}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ and $\mathcal{F}^*$, with values of 5.40×10^{-13} µs and 7.91×10^{-14} µs, respectively. Even though the time synchronisation precision $\Delta[t]$ under $\Phi_{\mathsf{h}}(\mathbf{x}[t], \mathbf{L}[t]; \mathbf{W})$ is slightly higher than that of the centralised controller, it effectively reduces communication overhead through the layered design of the distributed communication scheme. The conventional distributed controller exhibits slower convergence and yields a larger mean precision of 0.8437 µs, compared to both the proposed GNN-based control policy and the centralised controller. This may be because the fixed control gain, which is designed based on the known initial communication network topology, cannot adapt to frequent topology changes caused by the UAV mobility.

2) Performance Comparison among GNN-based Control Policies: We then compare our proposed distributed GNNs with the method in [31] and its variants. Here, we mainly focus on the GNN-based control policies, as a comparison between GNN-based and fully-connected neural network-based control policies has been conducted in [4]. The Graph Convolutional Neural Network (GCNN) with the readout phase is adopted

TABLE V
AVERAGE AND STANDARD DEVIATION OF $\sum_{t=0}^{T} c[t]$ FOR DIFFERENT TRAINED GNN MODELS.

	Normalised Laplacian matrix	**Normalised adjacency matrix**
GCNN w readout [31]	6.13 ± 1.58	6.23 ± 1.56
GCNN w/o readout	8.67 ± 2.26	9.27 ± 2.23
Dist. GNN w/ heat kernel	1.13 ± 0.14	1.09 ± 0.38
Dist. GNN w/o heat kernel	1.01 ± 0.13	0.78 ± 0.34

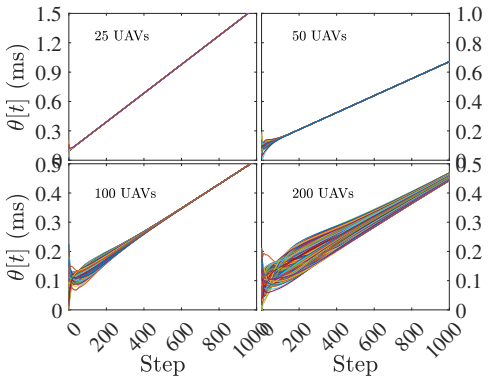

Fig. 4. Evolution of the clock offset under $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ for different swarm sizes.

from [31] and used as our baseline. This model consists of a single layer (i.e. $L = 1$) with $K = 3$ and $G_\ell = 64$, along with the 64-2 MLP-based readout. The `Tanh` activation function is also adopted in this GCNN model [31] [see Equ. (8)].

Table V shows the comparison results. It can be seen that our distributed GNNs outperform the GCNN model in [31]. From Equ. (4), time synchronisation in aerial swarms using the centralised controller is a linear system. By removing the activation function (i.e. `Tanh`), the proposed distributed GNN $\Phi_{\mathsf{l}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ without the heat kernel represents a linear parameterisation of the centralised controller. In this study, the GNN parameterisation of the controller aligns with the linear characteristics of the networked clock model [i.e. Equ. (2)]. Thus, our proposed GNN model $\Phi_{\mathsf{l}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ can achieve better synchronisation performance. This means that including a activation function may not necessarily be beneficial for improving GNN performance, and similar observations can also be found in [18].

From Table V, although $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ with the heat kernel performs slightly worse than $\Phi_{\mathsf{l}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$] without it, the latter uses more layers (i.e. $L = 4$) compared to the distributed GNN with the heat kernel (i.e. $L = 2$). Stacking more layers requires exchanging additional multi-hop UAV information, which increases the communication overhead. The distributed GNN $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ with the heat kernel achieves competitive performance under the same architectural settings, as indicated in Tables III and IV. In addition to leveraging the Laplacian matrix, we also train and test GNNs using the adjacency matrix. GNNs trained using Laplacian and adjacency matrices achieve similar time synchronisation performance.

3) Scalability Evaluation of the Proposed GNNs: Our GNN models, trained on swarms of 50 UAVs, are also evaluated under various swarm sizes, and the results are presented in Fig. 4 and Table VI. When the number of UAVs in the swarm is less than or equal to 200, the trained GNN models allow the UAV clocks to achieve time synchronisation during the simulations. As shown in Fig. 4, the convergence time increases with the swarm size. From Table VI, the costs associated with the GNN models are higher than those of the centralised controller, mainly due to the longer convergence times. Overall, without requiring re-training, the proposed distributed graph neural networks achieve synchronisation performance comparable to that of the centralised controller $\mathcal{F}^*$, even when scaled to larger swarm sizes, such as 200 UAVs.

However, our trained GNN models fail to achieve synchronisation during the simulations when the swarm size exceeds 400 UAVs. This may be due to the slow convergence speed (i.e. a large convergence time), as indicated in Fig. 4. Further investigation is needed to understand the reason behind this phenomenon.

C. GNN-based Resilient Synchronisation under Link Failures

In the literature, various types of jamming attacks have been studied, including constant jamming and random/periodic jamming attacks [6]. Even though these attacks differ in their interference signal transmission strategies, they share a common goal: to disrupt communication links either temporarily or permanently. Accordingly, to evaluate the resilience of our trained GNN models under changing swarm networks, we simulate the temporary and permanent communication link failures caused by jamming attacks, as shown in Fig. 5.

For Attack Mode 1, the circular attack zone remains fixed, while a swarm flies through it (see Fig. 5a). As a result, the swarm communication network topology is temporarily compromised. For Attack Mode 2, we continuously disrupt the communication among UAVs within a certain circular zone of an aerial swarm (see Fig. 5b). This means that the jamming attacks permanently change the wireless network topology. During the simulations, we vary the attack radius from $1.0\,\mathrm{m}$ to $2.5\,\mathrm{m}$ in increments of $0.5\,\mathrm{m}$.

Figs. 6 and 7 show the evolution of the clock offset and synchronisation precision $\Delta[t]$ under the proposed distributed GNN-based control policy $\Phi_{\mathsf{h}}\left(\mathbf{H}; \mathbf{L}, \mathbf{W}\right)$, the centralised controller $\mathcal{F}^*$ and the conventional distributed controller, in the presence of the temporary (i.e. Attack Mode 1) and permanent (i.e. Attack Mode 2) communication link failures, respectively. It is important to note that each subplot in Fig. 6 contains the offsets of all the 50 UAVs, which are also used to compute the precision $\Delta[t]$. In contrast, each subplot in Fig. 7 includes only the clocks of the UAVs that are not affected by jamming, and these are used to calculate $\Delta[t]$. In both attack modes and across different attack radii, our trained GNN models and the centralised controller are able to maintain clock synchronisation during the simulations. The synchronisation performance of the proposed GNN-based control policy remains comparable to that of the centralised controller under both the temporary and permanent communication link failures, without requiring re-training.

For the conventional distributed controller in Attack Mode 1 (i.e. temporary link failures; see Fig. 6), the low convergence

TABLE VI
AVERAGE AND STANDARD DEVIATION OF $\sum_{t=0}^{T} c[t]$ FOR DIFFERENT SIZES OF SWARMS.

Num of UAVs in Swarms	25	50	100	200
$\mathcal{F}^*$	0.32 ± 0.017	0.33 ± 0.018	0.34 ± 0.0092	0.34 ± 0.0079
$\Phi_{\mathsf{l}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$	0.36 ± 0.025	0.97 ± 0.091	10.59 ± 1.05	200.31 ± 39.00
$\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$	0.38 ± 0.039	1.15 ± 0.21	12.85 ± 2.76	234.20 ± 40.74

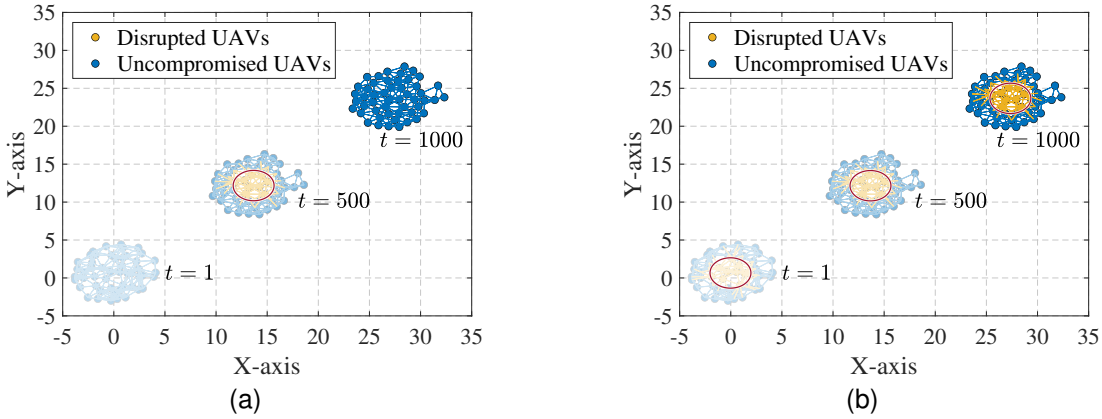

Fig. 5. Jamming attacks disrupt wireless communication among UAVs within a specific circular zone. (a) Attack Mode 1: the swarm flies through a fixed circular attack zone, leading to the temporary link failures. (b) Attack Mode 2: attacks continuously disrupt wireless communication among UAVs within a fixed circular zone, resulting in the permanent link failures.

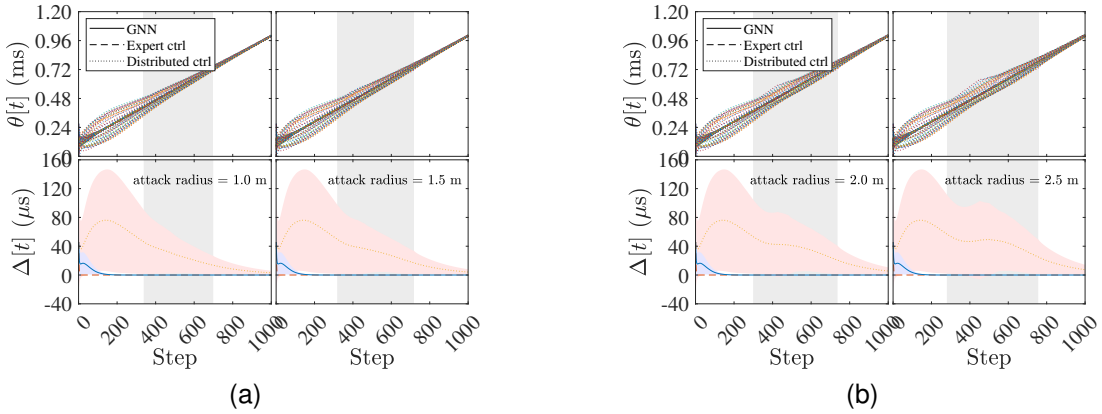

Fig. 6. Evolution of $\theta[t]$ and $\Delta[t]$ using the distributed GNN-based control policy $\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$, the centralised controller $\mathcal{F}^*$ and the distributed controller under the temporary communication link failures caused by Attack Mode 1. (a) Attack radius equals 1.0 (left) and 1.5 (right). (b) Attack radius is 2.0 (left) and 2.5 (right). The shaded regions indicate the periods during which the swarm experiences the temporary link failures.

speed results in poor synchronisation among the UAV clocks. During the attack period (i.e. the shaded regions in Fig. 6), the offsets drift continuously, and the precision increases slightly, as the clocks operate in free-running mode without achieving consensus on the skews. Once the disrupted communication links recover, the clocks can still be re-synchronised using the distributed controller. The same offset-drifting phenomenon is also observed under the GNN-based control policy. However, due to its high convergence speed, $\Phi_{\mathsf{h}}(\mathbf{H}; \mathbf{L}, \mathbf{W})$ allows all the clocks to rapidly achieve synchronisation, leading to a tiny precision value, as shown in Table VII. Even during the attack period, when the clocks are in free-running mode, the precision $\Delta[t]$ remains low owing to the consensus among, and the small magnitudes of, the clock skews. Furthermore, $\Delta[t]$ under the GNN-based control policy exhibits a decreasing trend throughout the simulations. This implies that the temporary communication link failures have little effect on the synchronisation performance, once the system reaches the steady synchronised state.

In Attack Mode 2, the permanent communication link failures in a subset of the swarm, namely, a fixed circular zone, leads to a more sparsely connected wireless network. This further degrades the convergence speed of the conventional distributed controller. For instance, as shown in Fig. 7, when the attack radius is $2.5\,\mathrm{m}$, the clock skews remain unsynchronised across the swarm, causing only a weak consensus trend in the offsets. Owing to the sparse swarm network topology in Attack Mode 2, the convergence speed under the GNN-based control policy is slightly reduced. However, clock synchronisation can still be achieved. The mean values of the

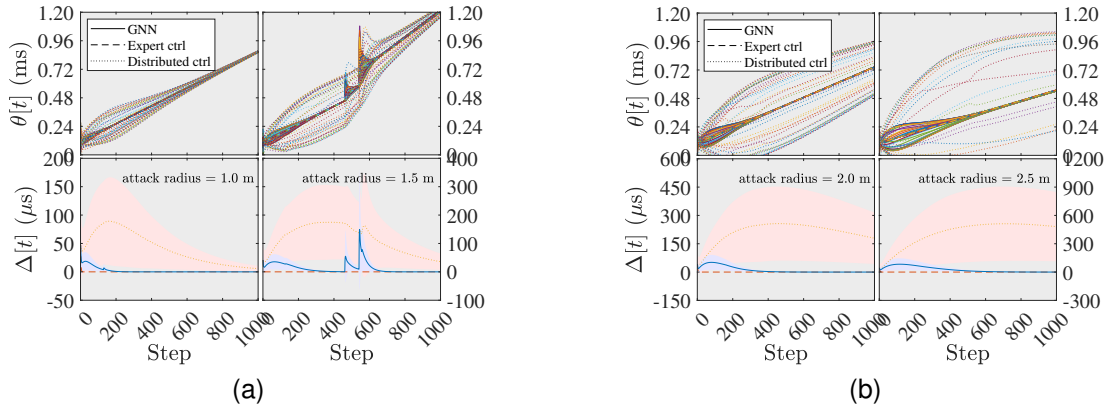

Fig. 7. Evolution of $\theta[t]$ and $\Delta[t]$ using the distributed GNN-based control policy $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$, the centralised controller $\mathcal{F}^*$ and the distributed controller under the permanent communication failures caused by Attack Mode 2. (a) Attack radius equals 1.0 (left) and 1.5 (right). (b) Attack radius is 2.0 (left) and 2.5 (right). The shaded regions represent that the swarm experiences the permanent link failures throughout the entire simulation.

TABLE VII
AVERAGE AND STANDARD DEVIATION OF $\Delta[t]$ OVER 100 STEPS BEFORE AND AFTER ENTERING/LEAVING THE ATTACK AREA FOR $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$ UNDER VARIOUS ATTACK RADII IN THE TEMPORARY COMMUNICATION LINK FAILURES CAUSED BY ATTACK MODE 1.

Attack radius [m]	$\Delta[t]$ **before entry** [µs]	$\Delta[t]$ **after entry** [µs]	$\Delta[t]$ **before exit** [µs]	$\Delta[t]$ **after exit** [µs]
1.0	$8.30 \times 10^{-3} \pm 8.00 \times 10^{-3}$	$1.23 \times 10^{-4} \pm 1.33 \times 10^{-4}$	$2.86 \times 10^{-7} \pm 3.66 \times 10^{-7}$	$1.63 \times 10^{-8} \pm 2.07 \times 10^{-8}$
1.5	$1.83 \times 10^{-2} \pm 1.75 \times 10^{-2}$	$3.37 \times 10^{-4} \pm 3.95 \times 10^{-4}$	$2.24 \times 10^{-6} \pm 4.26 \times 10^{-6}$	$1.82 \times 10^{-7} \pm 2.91 \times 10^{-7}$
2.0	$3.79 \times 10^{-2} \pm 3.63 \times 10^{-2}$	$8.26 \times 10^{-4} \pm 9.88 \times 10^{-4}$	$1.25 \times 10^{-5} \pm 2.38 \times 10^{-5}$	$1.06 \times 10^{-6} \pm 1.71 \times 10^{-6}$
2.5	$7.71 \times 10^{-2} \pm 7.40 \times 10^{-2}$	$1.90 \times 10^{-3} \pm 2.40 \times 10^{-3}$	$5.70 \times 10^{-5} \pm 1.13 \times 10^{-4}$	$5.64 \times 10^{-6} \pm 9.23 \times 10^{-6}$

TABLE VIII
MEAN AND STANDARD DEVIATION OF $\sum_{t=0}^{T} c[t]$ FOR GNNS TRAINED USING TWO DIFFERENT SCHEMES UNDER THE TEMPORARY COMMUNICATION LINK FAILURES CAUSED BY ATTACK MODE 1.

Attack radius [m]	**The proposed GNN policy** $\Phi_{\mathsf{h}}\left(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W}\right)$		**Expert control policy** $\mathcal{F}^*$	**Conventional distributed control**
	Adversarial imitation learning	**Imitation learning**		
0.0	1.13 ± 0.14	1.14 ± 0.15	0.33 ± 0.018	80.16 ± 12.90
1.0	1.59 ± 0.46	1.59 ± 0.45	0.34 ± 0.013	106.40 ± 24.24
1.5	3.96 ± 1.91	3.99 ± 2.03	0.36 ± 0.012	157.51 ± 55.82
2.0	16.20 ± 13.79	16.07 ± 13.55	0.39 ± 0.014	283.81 ± 142.55
2.5	52.18 ± 36.67	51.34 ± 35.63	0.45 ± 0.021	566.07 ± 368.42

precision $\Delta[t]$ over the last 100 steps (i.e. 1 s) under the four attack radii are 1.44×10^{-10} µs, 1.10×10^{-3} µs, 3.50×10^{-3} µs, and 9.60×10^{-2} µs, respectively. For the performance of $\Phi_{\mathsf{h}}\left(\mathbf{H}; \mathbf{L}, \mathbf{W}\right)$ when the swarm is split into two disconnected parts due to permanent communication failures, please refer to the Appendix.

Tables VIII and IX show that the GNN models under both training schemes are resilient against the communication link failures caused by two types of jamming attacks. Moreover, the GNN models trained with adversarial imitation learning [i.e. Equ. (19)] demonstrate slightly better performance than those trained with imitation learning. This is particularly clear when the attack radius is zero (i.e. no wireless link failures occur during testing) or under the permanent communication link failures (i.e. Attack Mode 2).

V. CONCLUSION

In this work, we have introduced the distributed graph neural networks $\Phi\left(\mathbf{H}; \mathbf{L}, \mathbf{W}\right)$ for synchronising clocks among UAVs. This approach is applicable to large swarms or those with dynamic wireless networks affected by the high mobility or the temporary and permanent wireless link failures.

We propose integrating the heat kernel into the GNN architecture, allowing it to retains low-frequency graph signals while attenuating high-frequency ones. This is consistent with the aim of time synchronisation, which is to ensure that the states of all the clocks are the same, corresponding to low-frequency graph signals. By setting K to 1, each UAV in the swarm is modelled as a node in our GNN architecture, where the one-hop feature aggregation is equivalent to the one-hop packet exchange between neighbouring UAVs. The swarm's distributed communication network is naturally represented through the GNN's message-passing scheme. Thus, stacking L layers allows each node to aggregate information from its L-hop neighbourhood, without requiring a fully-connected network topology. The proposed distributed GNNs $\Phi\left(\mathbf{H}; \mathbf{L}, \mathbf{W}\right)$ reduce communication overhead, when compared to state-of-the-art decentralised GNNs, such as those in [31].

Leveraging the adversarial imitation learning scheme, our GNN-based control policies achieve the similar synchronisation performance (without requiring re-training) when scaled to large swarms, as compared to the centralised controller using a fully-connected wireless network. Once trained, the proposed GNN-based control policies are also resilient to vary-

TABLE IX
MEAN AND STANDARD DEVIATION OF $\sum_{t=0}^{T} c[t]$ FOR GNNS TRAINED USING TWO DIFFERENT SCHEMES UNDER THE PERMANENT COMMUNICATION FAILURES CAUSED BY ATTACK MODE 2.

Attack radius [m]	**The proposed GNN policy** $\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$		**Expert control policy** $\mathcal{F}^*$	**Conventional distributed control**
	Adversarial imitation learning	**Imitation learning**		
0.0	1.13 ± 0.14	1.14 ± 0.15	0.33 ± 0.018	80.16 ± 12.90
1.0	1.93 ± 0.35	1.94 ± 0.33	0.33 ± 0.014	165.80 ± 22.13
1.5	5.70 ± 1.67	5.77 ± 1.85	0.33 ± 0.016	447.02 ± 66.25
2.0	15.78 ± 4.44	15.91 ± 4.73	0.33 ± 0.017	1123.84 ± 203.57
2.5	78.53 ± 39.65	76.42 ± 33.63	0.32 ± 0.016	3523.94 ± 704.55

ing wireless networks, including the temporary or permanent communication link failures, and can maintain synchronisation even when the swarm is split into two disconnected parts. Therefore, our proposed distributed graph neural networks can serve as a backup controller in safety-critical aerial swarms.

APPENDIX: SPLITTING SWARMS INTO TWO DISCONNECTED PARTS UNDER ATTACK MODE 2

For the permanent communication link failures in Attack Mode 2, we also study the performance of the proposed GNN-based control policy in the case where a swarm is split into two disconnected parts, as shown in Fig. 8. Fig. 9 shows the evolution of the clock offset, skew and precision under $\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$ and two traditional controllers in this scenario.

In Fig. 9, even though the swarm is split into two parts, our GNN-based control policy $\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$ still allows the clock offsets in each part to converge to consensus. A similar phenomenon is observed under the conventional distributed controller; however, its convergence speed is lower, and the clocks fail to achieve consensus during the simulations. For the centralised controller, despite the swarm being split into two parts, the UAVs in both parts can still communicate with the central UAV. Hence, all the UAV clocks in both parts achieve synchronisation. This is also reflected in the evaluation of $\Delta[t]$. The precision values under both the GNN-based control policy and the distributed controller increase, while the precision $\Delta[t]$ under the centralised controller remains near zero during the steady synchronised state.

Overall, even when the permanent communication link failures split a swarm into two disconnected parts, the proposed distributed GNN-based control policy $\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$ can still achieve resilient time synchronisation in each part.

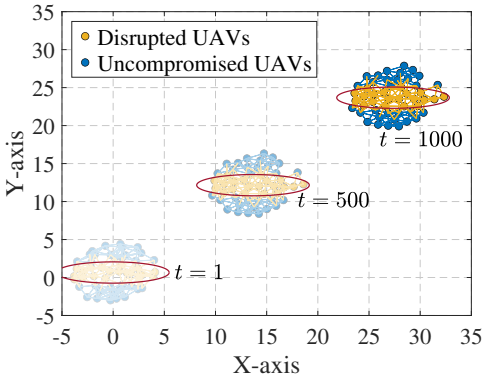

Fig. 8. Permanent communication link failures in Attack Mode 2 split the swarm into two disconnected parts.

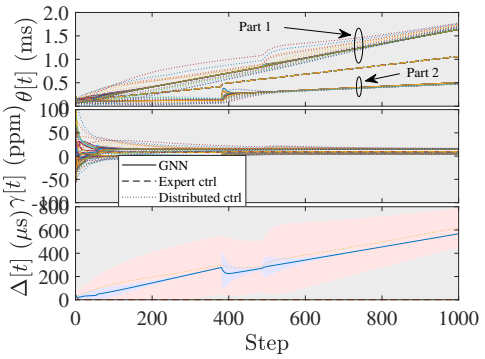

Fig. 9. Evolution of the clock offset, skew, and time synchronisation precision under $\Phi_{\mathsf{h}}(\mathbf{x}[t],\ \mathbf{L}[t];\ \mathbf{W})$ (solid line), the centralised controller $\mathcal{F}^*$ (dashed line) and the conventional distributed controller (dotted line), when the swarm is permanently split into two parts.

REFERENCES

[1] I. Bekmezci, O. Sahingoz and S. Temel., "Flying Ad-Hoc Networks (FANETs): A Survey," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 1254-1270, May 2013.

[2] K. Bai, J. Wu and H. Wu., "High-precision Time Synchronisation Algorithm for Unmanned Aerial Vehicle Ad Hoc Networks based on Bidirectional Pseudo-range Measurements," *Ad Hoc Netw.*, vol. 152, Jan. 2024.

[3] J. Gielis, A. Shankar and A. Prorok., "A Critical Review of Communications in Multi-robot Systems," *Current Robot. Reports*, vol. 3, pp. 213–225, Aug. 2022.

[4] F. Gama and S. Sojoudi., "Graph Neural Networks for Distributed Linear-Quadratic Control," *Proc. 3rd Conf. Learn. Dynamics Control*, Jun. 2021.

[5] Z. Zheng, A. Sangaiah and T. Wang., "Adaptive Communication Protocols in Flying Ad Hoc Network," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 136-142, Jan. 2018.

[6] H. Pirayesh and H. Zeng., "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 767-809, 2022.

[7] Y. Zong, S. Liu, X. Liu, S. Gao, X. Dai and Z. Gao., "Robust Synchronised Data Acquisition for Biometric Authentication," *IEEE Trans. Ind. Informat.*, vol. 18, no. 12, pp. 9072-9082, Dec. 2022.

[8] Y. Zong, X. Dai, Z. Wei, M. Zou, W. Guo and Z. Gao., "Robust Time Synchronisation for Industrial Internet of Things by H_∞ Output Feedback Control," *IEEE Internet of Things J.*, vol. 10, no. 3, pp. 2021-2030, Feb. 2023.

[9] R. Levie, W. Huang, L. Bucci, M. Bronstein and G. Kutyniok., " Transferability of Spectral Graph Convolutional Neural Networks," *J. Mach. Learn. Research*, vol. 22, pp. 1-59, Nov. 2021.

[10] J. Blumenkamp, S. Morad, J. Gielis, Q. Li and A. Prorok., "A Framework for Real-world Multi-robot Systems Running Decentralised GNN-Based Policies," *Proc. Int. Conf. Robot. Autom.*, May 2022.

[11] Y. Zong, N. Lu and B. Jiang., "Synthesising Controllers for Quadrotors via Imitation Learning," in *Proc. China Autom. Congr. (CAC)*, Nov. 2023.

[12] Y. Zong, X. Dai, P. Canyelles-Pericas, Z. Gao, W. NG, K. Busawon and R. Binns., "Synchronisation of Packet Coupled Low-accuracy RC Os-

cillator Clocks for Wireless Networks," *IEEE Trans. Wireless Commun.*, vol. 22, no. 7, pp. 4843-4857, Jul. 2023.

[13] J. Wu, K. Bai and H. Wu., "Advancing Convergence Speed of Distributed Consensus Time Synchronisation Algorithms in Unmanned Aerial Vehicle Ad Hoc Networks," *Drones*, vol. 8, Jun. 2024.

[14] X. Huan, K. Kim, S. Lee, E. Lim and A. Marshall., "A Beaconless Asymmetric Energy-Efficient Time Synchronisation Scheme for Resource-Constrained Multi-hop Wireless Sensor Networks," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1716-1730, Mar. 2020.

[15] H. Wang, L. Shao, M. Li, B. Wang and P. Wang., "Estimation of Clock Skew for Time Synchronisation Based on Twoway Message Exchange Mechanism in Industrial Wireless Sensor Networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4755-4765, Nov. 2018.

[16] K. Yildirim, R. Carli and L. Schenato., "Adaptive Proportional-Integral Clock Synchronisation in Wireless Sensor Networks," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 2, pp. 610-623, Mar. 2018.

[17] K. Xu, W. Hu, J. Leskovec and S. Jegelka., "How Powerful are Graph Neural Networks?" *Proc. Int. Conf. Learn. Representations* (ICLR), May 2019.

[18] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu and K. Weinberger., "Simplifying Graph Convolutional Networks," *Proc. 36th Int. Conf. Mach. Learn.* (ICML), Jun. 2019.

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu., "A Comprehensive Survey on Graph Neural Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[20] M. Defferrard, X. Bresson and P. Vandergheynst., "Convolutional Neural Networks on Graphs with Fast Localised Spectral Filtering," *Proc. 30th Int. Conf. Neural Inf. Process. Syst.* (NeurIPS), Dec. 2016.

[21] X. Wang and M. Zhang., "How Powerful are Spectral Graph Neural Networks," *Proc. 39th Int. Conf. Mach. Learn.* (ICML), Jul. 2022.

[22] T. N. Kipf and M. Welling., "Semi-Supervised Classification with Graph Convolutional Networks," *Proc. 5th Int. Conf. Learn. Representations* (ICLR), Apr. 2017.

[23] S. Segarra, W. Huang and A. Ribeiro., "Diffusion and Superposition Distances for Signals Supported on Networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 1, pp. 20-32, Mar. 2015.

[24] J. Ma, W. Huang, S. Segarra and A. Ribeiro., "Diffusion Filtering of Graph Signals and Its Use in Recommendation Systems," *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, Mar. 2016.

[25] F. Chung., "The Heat Kernel as the Pagerank of A Graph," *Proc. National Academy Sci. United States America* (PNAS), vol. 104, no. 50, pp. 19735-19740, Dec. 2007.

[26] B. Xu, H. Shen, Q. Cao, K. Cen and X. Cheng., "Graph Convolutional Networks using Heat Kernel for Semi-Supervised Learning," *Proc. 28th Int. Joint Conf. Artificial Intell.* (IJCAI), Aug. 2019.

[27] Q. Tan, Z. Wu, J. Lai, Z. Liang and Z. Ren., "HDGN: Heat Diffusion Graph Network for Few-Shot Learning," *Pattern Recognition Lett.*, vol. 171, pp. 61-68, Jul. 2023.

[28] J. Choi, S. Hong, N. Park and S. Cho., "GREAD: Graph Neural Reaction-Diffusion Networks," *Proc. 40th Int. Conf. Mach. Learn.* (ICML), Jul. 2023.

[29] J. Zhao, Y. Dong, M. Ding, E. Kharlamov and J. Tang., "Adaptive Diffusion in Graph Neural Networks," *Proc. 35th Conf. Neural Inf. Process. Syst.* (NeurIPS), Dec. 2021.

[30] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio., "Graph Attention Networks," *Proc. 6th Int. Conf. Learn. Representations* (ICLR), Apr. 2018.

[31] F. Gama, Q. Li, E. Tolstaya, A. Prorok and A. Ribeiro., "Synthesising Decentralised Controllers with Graph Neural Networks and Imitation Learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1932-1946, Apr. 2022.

[32] Q. Li, W. Lin, Z. Liu and A. Prorok., "Message-Aware Graph Attention Networks for Large-Scale Multi-robot Path Planning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5533-5540, Jul. 2021.

[33] S. Agarwal, A. Ribeiro and V. Kumar., "Asynchronous Perception-Action-Communication with Graph Neural Networks," arXiv:2309.10164, Sep. 2023.

[34] H. Ishii, Y. Wang and S. Feng., "An Overview on Multi-Agent Consensus under Adversarial Attacks," *Annual Rev. Control*, vol. 53, pp. 252-272, May 2022.

[35] J. Rewienski, M. Groth, L. Kulas and K. Nyka., "Investigation of Continuous Wave Jamming in An IEEE 802.15.4 Network," *Proc. 22nd Int. Conf. Microw. Radar*, May 2018.

[36] M. Wilhelm, I. Martinovic, J. B. Schmitt and V. Lenders., "Reactive Jamming in Wireless Networks—How Realistic is the Threat?" *Proc. 4th ACM Conf. Wireless Netw. Security*, Jun. 2011.

[37] Z. Chi, Y. Li, X. Liu, W. Wang, Y. Yao, T. Zhu and Y. Zhang., "Countering Cross-Technology Jamming Attack," *Proc. 13th ACM Conf. Security Privacy Wireless Mobile Netw.*, Jul. 2020.

[38] H. Kenlay, D. Thanou and X. Dong., "Interpretable Stability Bounds for Spectral Graph Filters," *Proc. 38th Int. Conf. Mach. Learn.* (ICML), Jul. 2021.

[39] L. Ruiz, L. F. O. Chamon and A. Ribeiro., "Transferability Properties of Graph Neural Networks," *IEEE Trans. Signal Process.*, vol. 71, pp. 3474-3489, Jul. 2023.

[40] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal and J. Tang., "Adversarial Attacks and Defenses on Graphs," *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 2, pp. 19–34, Jan. 2021.

[41] S. Gunnemann., "Graph Neural Networks: Adversarial Robustness," in *Graph Neural Networks: Foundations, Frontiers, and Applications*, L. Wu, P. Cui, J. Pei and L. Zhao, Eds., Singapore: Springer Singapore, 2022, pp. 149-176.

[42] L. Gosch, S. Geisler, D. Sturm, B. Charpentier, D. Zugner and S. Gunnemann., "Adversarial Training for Graph Neural Networks: Pitfalls, Solutions, and New Directions," *Proc. 37th Conf. Neural Inf. Process. Syst.* (NeurIPS), Dec. 2023.

[43] G. Giorgi and C. Narduzzi., "Performance Analysis of Kalman-Filter-based Clock Synchronisation in IEEE 1588 networks," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 8, pp. 2902-2909, Aug. 2011.

[44] L. Galleani., "A Tutorial on the Two-State Model of the Atomic Clock Noise," *Metrologia*, vol. 45, no. 6, pp. 175-182, Dec. 2008.

[45] C. Zucca and P. Tavella., "The Clock Model and Its Relationship with the Allan and Related Variances," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 52, no. 2, pp. 289-296, Feb. 2005.

[46] A. Ortega, P. Frossard, J. Kovacevic, J. Moura and P. Vandergheynst., "Graph Signal Processing: Overview, Challenges, and Applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808-828, May 2018.

[47] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar and A. Ribeiro., "Learning Decentralised Controllers for Robot Swarms with Graph Neural Networks," *Proc. Conf. Robot Learn.* (CoRL), Nov. 2020.

[48] D. Shuman, S. Narang, P. Frossard, A. Ortega and P. Vandergheynst., "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83-98, May 2013.

[49] A. Sandryhaila and J. Moura., "Big Data Analysis with Signal Processing on Graphs: Representation and Processing of Massive Data Sets with Irregular Structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80-90, Sep. 2014.

[50] Prorok Lab, "A Framework for Real-world Multi-robot Systems Running Decentralised GNN-based Policies," *YouTube*, Available: https://www.youtube.com/watch?v=COh-WLn4iO4, [Accessed: May 1, 2025].

[51] M. Balcilar, P. Heroux, B. Gauzere, P. Vasseur, S. Adam and P. Honeine., "Breaking the Limits of Message Passing Graph Neural Networks," *Proc. 38th Int. Conf. Mach. Learn.* (ICML), Jul. 2021.

[52] L. Ruiz, F. Gama, and A. Ribeiro., "Graph Neural Networks: Architectures, Stability, and Transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660-682, May 2021.

[53] R. Olfati-Saber, J. Fax, and R. Murray., "Consensus and Cooperation in Networked Multi-Agent Systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215-233, Jan. 2007.