Computer Communications 251 (2026) 108481

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

FALCON: Fanet-Aware Learning and digital twin CONtrol framework☆

Andrea Caruso [a,b,*], Christian Grasso [a,b], Raoul Raftopoulos [a,b], Giovanni Schembra [a,c]

[a] DIEEI, University of Catania - CNIT Research Unit, Catania, Italy
[b] CNIT Research Unit, Catania, Italy
[c] CNIT CONLab, Catania, Italy

ARTICLE INFO

Keywords:
Network digital twin
Smart agents
FANET
Cloud
Edge

ABSTRACT

The rapid evolution of telecommunication networks is leading to increasingly complex systems, requiring adaptive, flexible, and intelligent mechanisms for resource management, orchestration, and access control. In this context, the Network Digital Twin (NDT) paradigm emerges as a powerful tool to model the behavior of devices, communication links, operating environments, and applications in complex networks. This paper introduces FALCON, a Digital-Twin-based orchestration framework designed to optimize horizontal offloading in UAV-based Flying Ad Hoc Networks (FANETs) providing edge computing services to ground devices in remote areas. FALCON integrates multiple Smart Agents (DQN, A2C, PPO) running concurrently on the Digital Twin to dynamically determine the optimal offloading probabilities. A proof-of-concept demonstrates how the framework performs real-time What-if Scenario analyses and adapts to varying workload and channel conditions. Numerical results highlight the gains achieved through coordinated model selection and reuse, showing reduced end-to-end delay and faster convergence compared to standalone DRL-based controllers.

1. Introduction

Digital Twin (DT), originally rooted in manufacturing and industrial systems, has progressively evolved into a general paradigm for cyber–physical integration, where high-fidelity virtual replicas of physical assets are continuously synchronized through bidirectional data flows [1,2]. This idea has recently gained momentum in communication networks, where DTs are exploited to capture and manage the behavior of heterogeneous devices, communication links, and services under highly dynamic operating conditions.

Within communication networks, two closely related notions have emerged: Network Digital Twin (NDT) and Digital Twin Network (DTN). Although often used interchangeably, the two terms emphasize complementary perspectives that are useful to keep conceptually distinct.

A NDT denotes a DT of a given network, realized as a virtual replica of a physical network infrastructure (e.g., 5G/6G system or other complex networks). By this view, the network represents the twin target, around which telemetry, models, and control loops are organized to accurately mirror its state and behavior. Most architectural works [3–5] implicitly adopt this perspective: the NDT is a platform that ingests data from the physical network, maintains an up-to-date representation of it, and exposes this representation to analytics and automation functions.

A DTN, instead, shifts the emphasis on the network itself as an interconnected system of digital twins. Here, the network acts both as the object being mirrored and as the substrate interconnecting multiple twins of devices, services, and administrative domains. Thanks to this representation, a DTN enables cross-domain orchestration across RAN, transport, core, and edge segments. The DTN perspective is particularly visible in 6G-oriented visions, where each network function, slice, or service has its own twin, and these twins communicate to support global optimization and self-evolution.

In this work, we explicitly adopt the NDT perspective, creating a twin that mirrors and controls Flying Ad-hoc NETworks (FANETs). Compared to conventional terrestrial networks, FANETs are more dynamics: nodes are mobile, topologies change rapidly, wireless links are intermittent, and on-board computational and energy resources are constrained. These characteristics make FANETs an attractive yet challenging target for the NDT paradigm. Realizing a NDT for FANETs enables *what-if* analysis, policy exploration, and Machine Learning (ML) model training, reducing the risks and costs of experimentation on the physical aerial network.

In the envisioned scenarios, a FANET is treated as a self-consistent aerial infrastructure, operating without tight integration with terrestrial communication networks. The FANET provides services (for instance,

☆ This article is part of a Special issue entitled: 'DT Orchestration' published in Computer Communications.

* Corresponding author at: DIEEI, University of Catania - CNIT Research Unit, Catania, Italy.
E-mail addresses: andrea.caruso@phd.unict.it (A. Caruso), christian.grasso@unict.it (C. Grasso), raoul.raftopoulos@unict.it (R. Raftopoulos), giovanni.schembra@unict.it (G. Schembra).

https://doi.org/10.1016/j.comcom.2026.108481
Received 17 November 2025; Received in revised form 4 February 2026; Accepted 4 March 2026
Available online 6 March 2026

sensing, monitoring, or temporary connectivity) directly within the geographical area it covers by exploiting on-board communication, processing, and storage resources. In such isolated deployments, service performance and reliability primarily depend on the internal dynamics of the FANET, as described above, and a NDT for FANETs must therefore focus on faithfully capturing these phenomena to support the optimization of both network operation and service quality.

This setting naturally motivates a distributed realization of the NDT across the "edge-cloud" continuum of the FANET. Functions related to NDT management, including monitoring, synchronization, and ML-based policy enforcement, must be deployed close to the physical network to ensure low latency and real-time consistency. In contrast, computationally intensive tasks such as offline what-if analysis and training or re-training of complex ML models are delegated to a remote central cloud component associated with the FANET. Designing a NDT framework that partitions functionality between edge and cloud, while maintaining a consistent view of the FANET, is thus a key enabler for optimizing both network behavior and service performance.

With this in mind, this work proposes FALCON, a Fanet-Aware Learning and digital twin CONtrol framework, designed as a distributed architecture spanning edge and cloud, enabling the creation and management of a NDT for FANETs. Compared to the authors' previous work [6], the proposed framework introduces the following enhancements:

- FALCON is specifically designed for the management of FANET-oriented NDTs;
- the framework adopts a structured separation between an edge component, responsible for directly managing the NDT of the real FANET, and a cloud component hosting orchestration functionalities and the Smart Agents (SAs) used for what-if scenario analysis and model evaluation;
- the division into edge and cloud components enables FALCON to manage multiple FANETs simultaneously, even when they are deployed at different locations at the network edge, by replicating only the edge-side components while keeping the cloud-side components unique (apart from possible replication mechanisms). This design makes FALCON a scalable framework;
- the definition of communication interfaces between the edge-side and the cloud-side.

The rest of the paper is organized as follows. Section 2 reviews the state of the art on NDTs and their application to FANETs. Section 3 presents a system description, introducing FALCON and all its components. Section 4 describes the considered proof-of-concept and provides some numerical results that demonstrate the effectiveness of the proposed framework. Finally, Section 5 concludes the paper and outlines possible directions for future works.

2. State of art

This section reviews several works in the literature that address the topics covered in this paper. In particular, three main research directions have been identified: the first concerns the application of the NDT concept to 5G/6G communication networks; the second focuses on the use and implementation of basic digital twins for single UAVs or aerial networks; the third covers the first attempts at designing and deploying NDTs for FANETs.

Table 1 provides an overview of the main topics addressed by the works discussed in the following subsections, offering a comparison with what is proposed by FALCON.

2.1. Network digital twins for 5G/6G communication systems

Regarding the use of NDTs for 5G/6G communication systems, Zhou et al. in [3] provide the most widely accepted reference definition of an NDT as a digital representation of a communication network, tightly synchronized with the physical elements and used for analysis, decision support, and control. They define three main layers: the physical twin (real network), the virtual twin (model and state), and the synchronization mechanisms between them, plus functional blocks for data collection, modeling, analytics, and actuation.

Authors in [7] propose a DT for 5G networks, focusing on a modeling and deployment methodology. Compared to the previously cited work, they explain how to construct data pipelines, models of topology and services, and a deployment strategy using Software Defined Networking (SDN) and Network Function Virtualization (NFV) technologies. They build a network-centric twin: it captures the service graph, slices, and performance indicators. However, its use is mostly for offline experimentation and what-if analysis, and it requires substantial work to adapt to different network scenarios.

Mozo et al. with B5GEMINI [4] take a more automation-oriented perspective by proposing an Artificial Intelligence (AI)-driven NDT for 5G and beyond transport networks. In contrast to [7], which focuses on methodology and modeling, B5GEMINI emphasizes a microservice-based architecture and the integration of Machine Learning (ML) models directly into the twin. The NDT continuously ingests telemetry from the transport network, reconstructs the logical state, and runs ML-based prediction and anomaly detection modules.

Khan et al. [8] discuss digital-twin-enabled 6G as a vision wherein DTs of radio, transport, and edge segments cooperate to support stringent performance and massive heterogeneity. Their work is more high-level but stresses architectural trends, such as distributed twins at the edge and in the cloud, that later implementations of NDT start to adopt.

Recent works further specialize NDTs around particular use cases and modeling choices. Authors in [9] describe a network digital twin environment for 5G V2X systems, where the twin is used to predict traffic congestion, optimize traffic-light timing, and improve safety in a vehicular network. Their approach extends the NDT concept from transport/core networks to a joint communication-mobility system. They couple the NDT with vehicular mobility patterns, highlighting that NDTs can be used to co-design networking and domain-specific control (e.g., traffic control).

All these works assume relatively structured, mostly terrestrial networks, with very moderate topology dynamics. Compared with the FALCON framework, none deals explicitly with the extreme mobility, highly dynamic connectivity, and self-organized routing typical of FANETs. This aspect represents a motivation to extend the NDT concept to aerial ad hoc networks.

2.2. Digital twins for UAVs, swarms, and aerial networks

A second line of research applies DT concepts to UAVs, usually with a focus on control, mission planning, or environment modeling.

Lei et al. [10] use a DT to enable intelligent cooperation in UAV swarms by combining DT and ML. Their twin captures UAV dynamics, mission tasks, and some communication constraints, and is leveraged to design ML-based cooperation mechanisms (e.g., for coordinated flight and resource sharing).

Shen et al. [11] extend this paradigm by employing a digital twin of multi-UAV systems as a training environment for deep reinforcement learning controllers designed for flocking. This control-oriented twin enables safety-guaranteed flocking and demonstrates how digital twins can reduce both the risk and cost associated with training RL policies directly on real systems.

In the context of 6G green IoT, Qi et al. in [12] propose a UAV digital twin for wireless channel modeling, where the DT reconstructs 3D environments and UAV trajectories to predict channel conditions

Table 1
Comparison between state of the art and FALCON framework.

Topic	[3]	[4]	[7]	[8]	[9]	[10]	[11–13]	[14]	[15–17]	FALCON
NDT	x	x	x	x	x				x	x
FANET-oriented or UAV-oriented						x	x	x	x	x
Continuous P2V synchronization	x	x		x	x	x	x			x
Edge-Cloud structural split				x	x					x
Scalable multi-network management						x		x		x
Parallel Smart Agent, model selection and model reuse						x				x
Intent-driven orchestration	x			x						x

in a space-air-ground integrated network. This allows accurate channel prediction and supports link adaptation and resource allocation policies.

Valencia et al. [14] propose an open-source UAV digital twin framework validated on remote sensing in the Andean Mountains. Their twin integrates ROS-Gazebo and Ardupilot, enabling realistic reconstruction of environments and UAV platforms. This framework emphasizes engineering aspects: reproducibility, modularity, and open-source tools that can be reused across applications. While it focuses on sensing rather than networking, it illustrates how DT platforms can be structured as reusable software infrastructure for multi-UAV missions.

Finally, Li et al. in [13] consider UAV-aided edge networks with Ultra-Reliable and Low-Latency Communications (URLLC) and propose a digital twin approach to manage them. Their DTs model both UAV dynamics and the edge network, including latency and reliability requirements, to control offloading and resource allocation for URLLC services.

While these DTs are starting to be mission- and network-aware components, these works do not realize a complete network digital twin of a FANET: they typically assume infrastructure-assisted networks (UAV + edge/5G) or focus on specific aspects such as channel shaping, without explicitly modeling and controlling the whole FANET down to the application layer.

2.3. Towards network digital twins of FANETs

The explicit use of DT as network-level replicas for flight networks and FANETs is still under development, but several contributions point in this direction.

Moorthy et al. [15] propose a middleware for DT-based flight network simulations, integrating UBSim (for drone dynamics) and UBANC (for aircraft networking). Their middleware maintains a virtual representation of a flight network, continuously fed with simulated telemetry data, and supports the evaluation of communication protocols and applications under realistic mobility and channel conditions. This work addresses the joint modeling of mobility and communication, although primarily in an offline simulation context. It implements fundamental NDT mechanisms, synchronization interfaces, shared state, and controllable what-if scenarios, without yet being integrated with a real FANET.

Cauwe et al. in [16] present GazeboNS3, a DT system for UAV swarms that tightly couples the Gazebo physical simulator with the ns-3 network simulator. This enables richer co-emulation of physical and network behavior and facilitates integration with existing control stacks. It is an excellent testbed for algorithms and protocols, but does not yet provide full online synchronization with real UAV networks as an operational NDT would.

El-Basioni [17] takes a decisive step towards NDT for FANETs by proposing a data-driven DT approach for the co-design of routing, topology, and mobility in FANET systems. In this context, UAV and link telemetry are used to construct a FANET twin, including node locations, link quality, and traffic patterns. A model-based RL algorithm is run on the twin to explore strategies that jointly adapt UAV routing and trajectories. The resulting policies are then applied to the real FANET. This creates a true network-centric twin: the primary object of the DT is the FANET, and the twin is leveraged for safe exploration and optimization.

Compared with these works, the FALCON framework goes beyond single-scenario, single-twin designs. It is built as a generic management framework for multiple FANET NDTs, each potentially served by different SAs and optimization strategies. Furthermore, FALCON targets operational NDTs directly attached to live FANETs at the edge, with a clear separation of concerns between edge-side NDT maintenance and cloud-side orchestration and learning.

3. System description

In this paper, we consider a network orchestration platform, named FALCON, for the management of a FANET by means of the Digital Twin paradigm. A FANET represents the physical twin (FPT: FANET Physical Twin) to be managed, while its digital twin, usually referred to as Network Digital Twin (NDT), hereafter will be named FANET Digital Twin (FDT).

The objective of FALCON is to manage FANETs to ensure the continuity of the provided service while addressing the specific requirements associated with that service. Each FANET is composed by a number of UAVs, connected to each other with full line-of-sight or through multi-hop routes. In line with the paradigm of the network digital twin, when a FANET uses the FALCON platform to be managed, a digital replica of it is created, that is, the FDT.

The FALCON platform has the objective of deciding, for each FANET, a specific optimization policy, based on machine learning, to be applied at runtime. This is achieved by evaluating and comparing different policies simultaneously, which are applied on the digital representation of the FANETs, i.e. without the need of acting directly on the physical network. The architecture of the proposed FALCON framework is presented in Fig. 1.

It is distributed in the edge-to-cloud continuum, with the following two components:

- the *FALCON Edge*, that is the FALCON component that is deployed at the edge of the network, very close to the FANETs to be managed. It contains the FDTs, and is responsible for interacting with the relevant FPTs;
- the *FALCON Cloud*, placed on the other side of the core network with respect to the physical FANETs, i.e. in the cloud. It is in charge of running different AI-based decision making policies in order to compare them at runtime, and identify the one that best fits the intents received from external users taking into account the current status of the physical networks.

In the following subsections, these two components will be described in detail, including their internal architecture and data flows.

3.1. FALCON Edge

The FALCON Edge component of the proposed framework is located close to the physical network devices that will be managed. In the figure, F FANETs are depicted, each deployed on a specific geographic area, at the edge of the network.

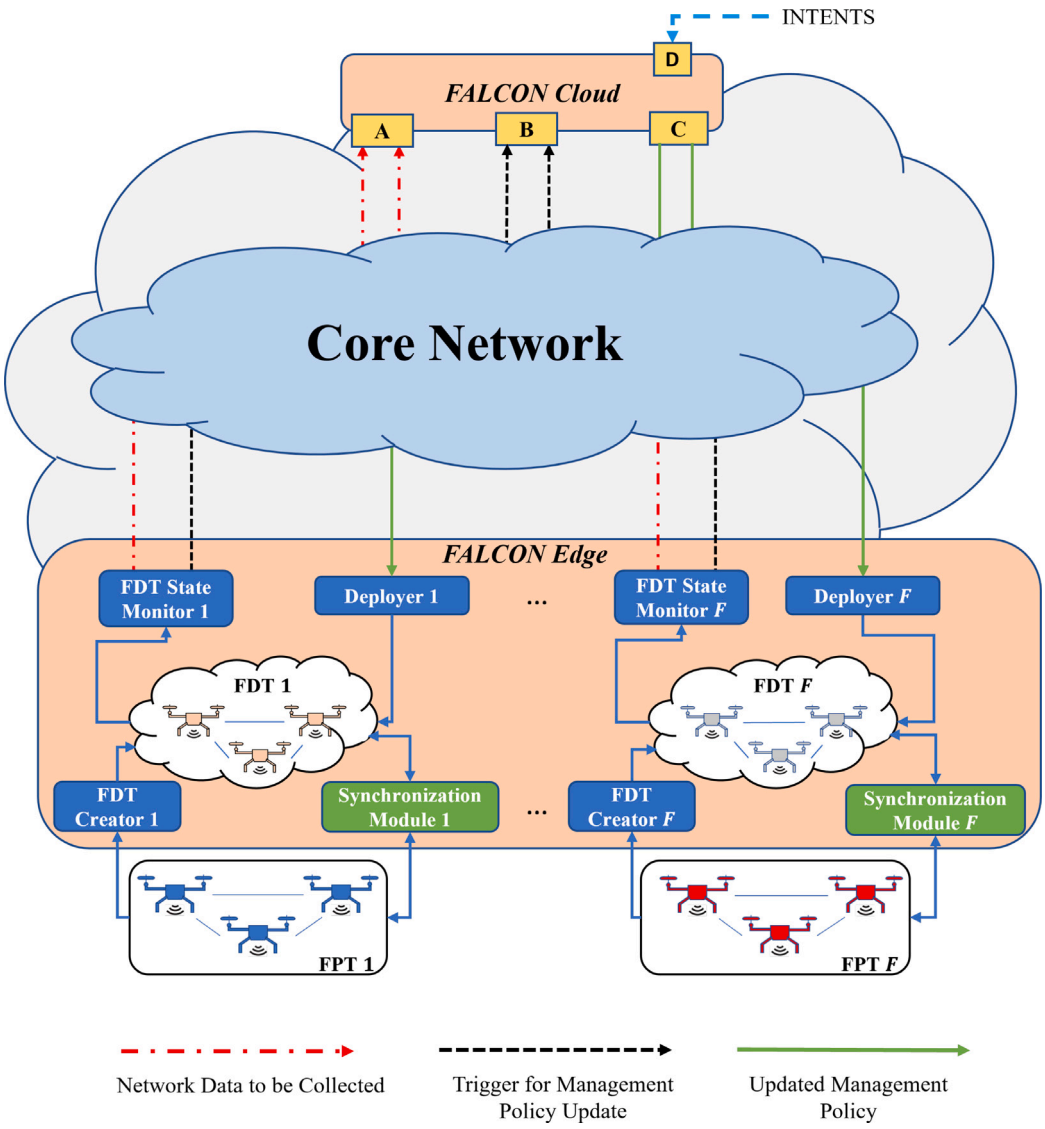


Fig. 1. FALCON Edge architecture.

Each FDT is not only the digital representation at the infrastructure level (e.g. number of UAVs and communication channels between them), but also represents the FPTs in terms of the data generated by each UAV, including both telemetry data and application-level data related to the service running in the network.

The creation of the FDT is performed by the *FDT Creator* block, within the FALCON Edge. One FDT Creator is assigned to each FPT. Its task is to receive the initial state of the FPT and build a virtual environment in which the FDT can be instantiated. The choice of the virtualized technique to be used (virtual machines or containers/dockers) depends on the application and on the available virtualization framework, and is out of the scope of this work. To ensure that each FDT accurately reflects its physical counterpart in real time, low latency and very fast data exchange are needed due to the high mobility and high-frequency topology changes of a typical FANET scenario. This is the reason why the FDTs are deployed at the edge of the network, very close to the physical networks.

To guaranty continuous synchronization, another component of the FALCON Edge is the *Synchronization Module*, one for each FANET. Its role is to implement the Physical-to-Virtual (Ph2Vi) pipeline between physical and virtual devices, ensuring that every component of the FDT is an accurate replica of its corresponding physical element along the time. The frequency at which data are received from the FPT and forwarded to the FDT through this module depends on the application scenario and on the dynamics of the network topology. More in deep, changes in network topology are due to the instantaneous node availability that is time-variant because UAVs are battery-powered, so each one may become unavailable when it needs to change its battery, or returns back after battery replacement.

On the other hand, to ensure that the FALCON Cloud has the needed information on the current FANET status to identify the most suitable policy, the FALCON Edge contains, for each FANET, a *FDT State Monitor*. This module is responsible for monitoring all events that occur within the PDT to which it is connected.

All collected information, ranging from the telemetry data of each UAV to the status of the interconnection links and the application-level data, is aggregated and sent to the FALCON Cloud through two dedicated input interfaces: *interface A* and *interface B* (see Fig. 1). These interfaces are designed to receive information from all the FDT State Monitors.

Finally, another block of the FALCON Edge is the *Deployer*, one per FANET. Its main task is to apply, on the corresponding FDT, the directives received through the *interface C* from the FALCON Cloud, namely the AI-based model that, based on the network state and the Intents, has been evaluated as the most suitable among the available ones.

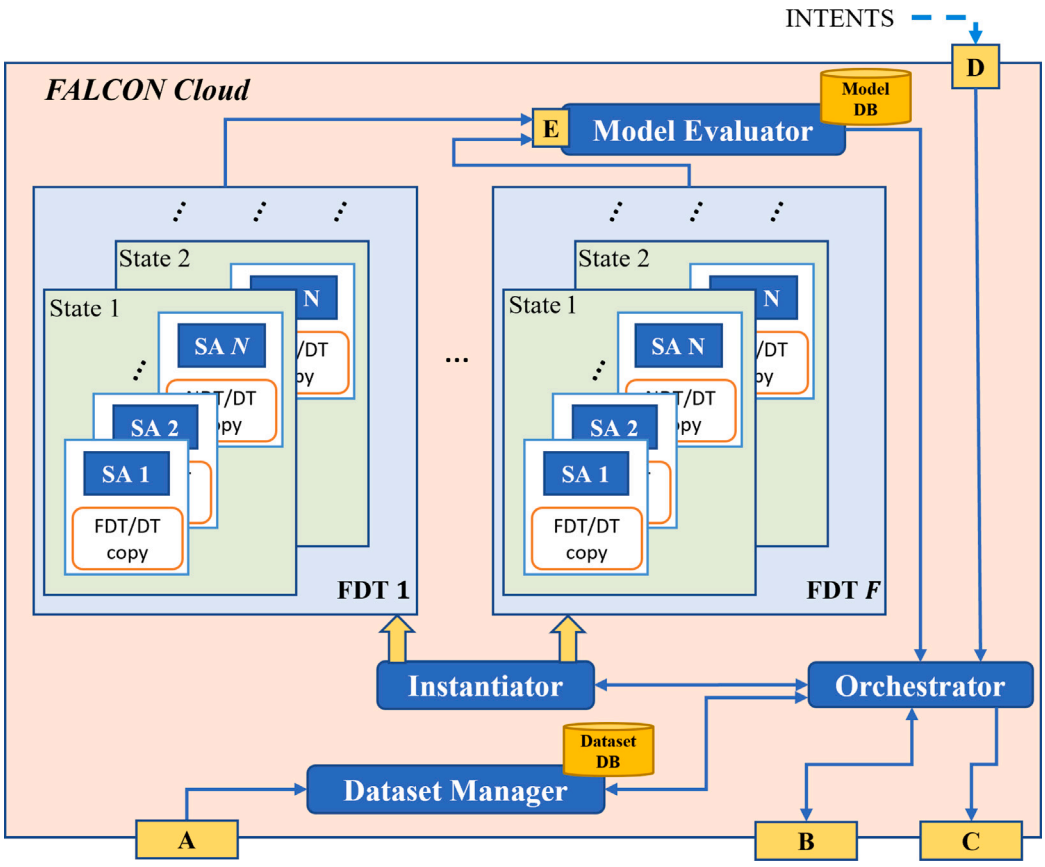


Fig. 2. FALCON Cloud architecture.

3.2. FALCON Cloud

The FALCON Cloud, whose architecture is shown in Fig. 2, runs on a remote cloud and communicates with the FALCON Edge through the Core Network. It is primarily responsible for deploying and testing different AI-based models to detect the one that best suits the network in a given state. Unlike the components located in the edge part, which are instantiated once for each of the FANETs managed by the framework, in the FALCON Cloud there are single instances for the entire system, although backup mechanisms can be implemented to avoid failures impacting the whole framework.

The core component of FALCON Cloud is the *Orchestrator*, whose central role within the framework is to coordinate the operation of all the other modules to achieve an architecture capable of efficiently managing all physical elements through the decisions made on their digital counterparts. The efficient management of each FPT is driven by the *Intents* that the Orchestrator receives from an external entity (for example, a service provider or a user requiring specific service guaranties of an application deployed through the FANET). All Intents are communicated to the Orchestrator using *Interface D*, which is exposed externally by the FALCON Cloud.

After receiving the Intent for an FPT, the Orchestrator needs to know the current FANET state to implement the best policies to satisfy the requirements. To do this, it communicates with the *Dataset Manager*. This component uses a *Dataset DB* in which all the data received from the FDT State Monitor through the *Interface A* are processed, organized, and stored in the form of datasets.

The behavior of the Orchestrator is not triggered solely by the reception of Intents, but also by alerts sent by the FDT State Monitors and delivered directly through the *Interface B* (see Fig. 2). This makes the system significantly more responsive to changes in the state of the FANETs (which, as noted so far, are highly dynamic networks), allowing prompt intervention whenever adjustments in the adopted decisions are required.

After the reception of a new Intent and the need to modify the network behavior because not compliant with the desired one, the Orchestrator triggers the *Instantiator*. This component has the goal of running multiple instances of SAs, each capable of executing a specific ML technique or optimization tool, such as, for example, Integer Linear Programming (ILP) or time-series forecasting methods based on Long Short-Term Memory (LSTM). Each SA aims to optimize a specific policy and produce a model for the management and orchestration of the FDT under study. To achieve this, when an SA is launched, the Orchestrator must provide it with a copy of the FDT state retrieved from the Dataset Manager. From this virtual replica of the physical network, each SA applies its technique to obtain the best possible outcome.

The Instantiator can manage multiple SA instances simultaneously, depending on how many policies need to be evaluated and compared. If, during the execution of these SAs, the FANET enters a new state, the Instantiator can launch a new set of SAs, working on the updated representation of the FDT. However, the previously active SAs will not be stopped; instead, they will continue their evaluation until their respective goals are reached. This allows FALCON to maintain ready-to-use models for future situations in which a FANET may re-enter a state that has already been analyzed.

Each time an SA completes its task, it sends its final results to the *Model Evaluator* through *Interface E*. The Model Evaluator analyzes the output produced by the SA and stores the best-performing model in the *Model DB*. When another SA later contacts the Model Evaluator, the latter compares the newly received output with the model already stored in the Model DB for the same FDT and the same initial network state. If the new model outperforms the existing one, it is replaced in the database; otherwise, it is discarded.

Whenever a new model needs to be applied to an FDT, the Orchestrator searches in the Model DB for the current state of the FDT: if it is

present in the database, the Orchestrator retrieves the best performing model directly and sends it, via the *Interface C* of the FALCON Cloud, to the Deployer in the FALCON Edge. As described in the previous subsection, the Deployer is in charge of actuation of this model within the FDT.

If no suitable model is available in the database, the Orchestrator will trigger the Instantiator to create a new one.

4. Proof of concept and numerical results

The FALCON framework introduced in this paper and described in the previous section is designed to be used to optimize any FANET optimization problem. In this section, as a use case, we show how FALCON can be exploited to optimize the horizontal offloading problem among UAVs in a FANET, in order to minimize the job processing delay in that FANET. More specifically, Section 4.1 describes the considered setup, Section 4.2 introduces the SAs used in the PoC, while Section 4.3 presents some numerical results.

4.1. PoC description

Let us consider a FANET designed to provide edge computing services to a remote geographic area [18,19]. More specifically, devices deployed on the ground use the edge computing facilities of the FANET to offload computation jobs that cannot be processed locally. The covered area is divided into multiple zones, each served by one of the FANET UAVs.

The job offloading process in each zone alternates between different activity states, leading to periods of overload and underload for each UAV. Therefore, each UAV, upon receiving each job from ground devices, can decide whether to process it locally or offload it to another UAV by means of horizontal offloading. The UAV that receives a job from a ground device is called the Dwell UAV for that job, while the UAV that actually processes the job is referred to as its Processing UAV [20]. The offloading decision taken by the Dwell UAV is performed for each job considering the delay the job would suffer in the local Processing Queue if it were not offloaded to another UAV, or in both the Offloading Queue of the Dwell UAV and in the Processing Queue of the Processing UAV if it were offloaded. The objective is to minimize the average latency. This decision is taken according to a given offloading probability, which plays a crucial role in the system's performance, and requires to be optimized.

In this article, we consider a FANET made up of four UAVs, and we assume that the zone covered by each UAV alternates between two different activity states, referred to as low-activity (L) and high-activity (H) states.

The FDT associated to each FANET, behaving as the relevant FPT, is run on either a server at the network edge (a ground server), or at the far edge, i.e. mounted onboard of one of the UAVs in the same FANET. From a scalability perspective, FALCON adopts a FANET-level replication model, where each FANET is associated with an independent FDT instance deployed at the edge or far edge. This design results in linear scalability with respect to the number of FANETs, while avoiding cross-FANET synchronization and centralized bottlenecks.

An FDT comprises the DT of all the UAVs in the FANET, connected through a digital representation of the wireless links among the physical UAVs. The representative information exported through the Synchronization Module by the FANET includes:

- for each UAV, the number of jobs waiting in the Processing and the Offloading Queues;
- for each link, the Signal-to-Interference-plus-Noise Ratio (SINR), the power spectrum density of the noise and the available bandwidth for data transmission (parameters that allow the FALCON Edge to calculate the available transmission rate of that link).

The FDTs provide these data to the FDT State Monitor. In the reverse direction, the Deployer module sets the offloading probabilities on all UAVs of the FDT, which are synchronized with the physical UAVs at runtime.

Synchronization between the FPT and the FDT is bidirectional and asymmetric. The synchronization flow from the FPT to the FDT is continuous and consists of lightweight telemetry data describing UAV status, queue occupancies, and link conditions. Due to the limited size and low bitrate of this information, continuous streaming is feasible and does not introduce any scalability bottlenecks. In the opposite direction, from the FDT to the FPT, synchronization is event-driven and limited to FANET configuration and control messages. These messages are generated only when a new orchestration decision is required, either due to new intents received by the Orchestrator or to significant changes in the FANET operating state detected at the edge. Control actions therefore operate on a service orchestration time scale of seconds or minutes.

As a consequence, FALCON enforces a soft-consistency model between the FDT and the FPT. This level of consistency is sufficient for service-level orchestration, what-if analysis, and policy evaluation, since synchronization does not constitute a bottleneck: the FPT-to-FDT data flow relies on continuous low-bitrate telemetry, while the FDT-to-FPT data flow is limited to event-driven configuration messages generated only when a new orchestration policy is deployed.

It is worth noting that the proposed framework does not introduce additional energy consumption on the UAVs, since all DT instances and SAs are executed on edge or cloud infrastructure, while UAVs are only involved in lightweight telemetry reporting and event-driven configuration updates.

4.2. Smart agents used by the FALCON Cloud in the PoC

To determine the best set of offloading probabilities to be assigned to the UAVs that minimizes the FANET delay, each SA uses the inverse of the mean delay, experienced by all the jobs in the system, as a reward.

Let $\mathcal{J}_n$ be the set of active jobs and $d_j(n)$ the delay of job j at each decision epoch n. The per-epoch reward is then defined as:

$$r_n = -\frac{1}{|\mathcal{J}_n|} \sum_{j \in \mathcal{J}_n} d_j(n) \tag{1}$$

The corresponding discounted return is:

$$G_n = \sum_{k \geq 0} \gamma^k r_{n+k} \tag{2}$$

where $\gamma \in [0, 1)$ is the discount factor. The action in epoch n represents the offloading probabilities applied to the UAVs in the FDTs.

At each decision epoch n, the state of each agent, denoted s_n, includes three components: (i) Processing-queue state K_n^{proc}, i.e., the queue lengths associated with each UAV's local processor; (ii) Offloading-queue state K_n^{off}, i.e., the queue lengths in each UAV's offloading buffer; and (iii) Zone-activity state ζ_n, i.e., the current activity level (e.g., low or high) of each zone, which determines the job arrival process. Hence, the state of the global system is expressed as $s_n = \left(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n\right)$.

Multiple SAs can run simultaneously, one for each policy to be evaluated and compared. In this paper, we will consider a Double Deep Q-Network (DQN) SA, an Advantage Actor Critic (A2C) SA, and a Proximal Policy Optimization (PPO) SA. In the following, we explain in detail how each of these SAs is implemented.

The *DQN* SA approximates the optimal action-value function $Q^\star(s, a)$ with a neural network $Q_\theta(s, a)$, using a target network Q_{θ^-} and a replay buffer. For each sampled transition $(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n, a_n, r_n, K_{n+1}^{\text{proc}}, K_{n+1}^{\text{off}}, \zeta_{n+1})$, the target value and loss function are given by:

$$y_n = r_n + \gamma \max_{a'} Q_{\theta^-}(K_{n+1}^{\text{proc}}, K_{n+1}^{\text{off}}, \zeta_{n+1}, a') \tag{3}$$

$$\mathcal{L}_{\text{DQN}}(\theta) = \mathbb{E}\left[(y_n - Q_\theta(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n, a_n))^2\right] \tag{4}$$

During training, each agent selects actions using an ϵ-greedy strategy on $Q_\theta(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n, \cdot)$, and periodically updates $\theta^- \leftarrow \theta$

The *A2C* SA learns a stochastic policy $\pi_\theta(a_n \mid K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)$ directly, assisted by a critic $V_\omega(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)$ that estimates the state value at decision epoch n. A2C introduces a term called advantage, that quantifies how much better (or worse) a specific action a_n is compared to the average cumulative reward expected at the same state $K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n$. The advantage is formally described by the following formula:

$$A_n = \left(r_n + \gamma V_\omega(K_{n+1}^{\text{proc}}, K_{n+1}^{\text{off}}, \zeta_{n+1})\right) - V_\omega(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n) \tag{5}$$

The corresponding loss functions for actor and critic, used to train the neural networks, are:

$$\mathcal{L}_{\text{actor}}(\theta) = -\mathbb{E}\left[\log \pi_\theta(a_n \mid K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)\, A_n\right], \tag{6}$$

$$\mathcal{L}_{\text{critic}}(\omega) = \mathbb{E}\left[\left(r_n + \gamma V_\omega(K_{n+1}^{\text{proc}}, K_{n+1}^{\text{off}}, \zeta_{n+1}) - V_\omega(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)\right)^2\right] \tag{7}$$

Finally, the *PPO* SA tends to learn more stable and robust policies by employing a clipped surrogate objective. In PPO SA, we define the probability ratio as a measure of how much the new policy changes the likelihood of taking action a_n compared to the old policy. This is defined by the following equation:

$$\rho_n(\theta) = \frac{\pi_\theta(a_n \mid K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)}{\pi_{\theta_{\text{old}}}(a_n \mid K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)} \tag{8}$$

Moreover, PPO uses an advantage estimate A_n derived from states $(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n)$ and $(K_{n+1}^{\text{proc}}, K_{n+1}^{\text{off}}, \zeta_{n+1})$, i.e., s_n and s_{n+1}, in the same way as in the A2C SA. To ensure that the new policy cannot deviate too far from the old one, thus stabilizing training, the target in the PPO SA is clipped between $1-\epsilon$ and $1+\epsilon$, as described in the following:

$$\mathcal{L}_{\text{clip}}(\theta) = \mathbb{E}\left[\min\Big(\rho_n(\theta)\, A_n,\ \text{clip}\big(\rho_n(\theta),\ 1-\epsilon,\ 1+\epsilon\big)\, A_n\Big)\right] \tag{9}$$

The overall PPO loss, used for training the neural networks combines this surrogate with value and entropy terms:

$$\begin{aligned}\mathcal{L}_{\text{PPO}}(\theta, \omega) = &- \mathcal{L}_{\text{clip}}(\theta) + \lambda_V\, \mathbb{E}\left[(r_n - V_\omega(K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n))^2\right] \\ &- \lambda_H\, \mathbb{E}\left[\mathcal{H}(\pi_\theta(\cdot \mid K_n^{\text{proc}}, K_n^{\text{off}}, \zeta_n))\right]\end{aligned} \tag{10}$$

where r_n is the empirical return at decision epoch n, and λ_V, and λ_H are weighting coefficients controlling the contribution of the value and entropy terms, respectively.

In the next section, we present the numerical results obtained in the considered PoC, highlighting how the different SAs evolve over time, how quickly they converge, and how effectively they react to changes in network conditions.

4.3. Numerical results

This section presents the numerical evaluation of the proposed DT-based orchestration framework in a representative FANET scenario. The objective is to assess how the coordinated operation of multiple SAs improves system responsiveness and service delay under varying network conditions, compared with standalone ML-based orchestrators. These results implicitly account for the effects of synchronization delays and DT-FANET consistency, as the Smart Agents operate on state information obtained through periodic synchronization between the physical network and its digital twin. The experiment emulates the time evolution of a geographic area monitored by a FANET in 2000 s. More specifically, we assume that each monitored area is characterized by an alternation of three distinct operating phases (Conditions C1, C2 and C3), thus testing the agents' ability to adapt to both workload and channel dynamics.

At time $t_1 = 0$ s, the system operates under Condition C1. In this state, the job-offload process from each zone follows a Poisson distribution with mean rates of 1.3 kjob/s during low-activity periods and 2.3 kjob/s during high-activity periods. On each UAV, the job-processing and offloading times are also Poisson distributed, with rates $\mu_P = 1.75$ kjob/s and $\mu_O = 3.5$ kjob/s.

After 600 s (at $t_2 = 600$ s), the system transits to Condition C2, which is characterized by a lower job complexity and consequently a higher job-processing rate, $\mu_P = 2.5$ kjob/s. Later, at $t_3 = 1000$ s, the system returns to C1 and remains there for 400 s, until $t_4 = 1400$ s. At that time, it switches to Condition C3, featuring an enhanced communication channel among UAVs and therefore an increased offloading rate of $\mu_O = 4.5$ kjob/s.

Throughout the entire experiment, all four SAs run concurrently within the FDT Manager. As depicted at the top of Fig. 4, whenever the FDT State Monitor detects a change in system condition, the Instantiator deploys a new set of SA instances, while the previous ones continue to operate until they reach convergence. Each SA label includes a subscript indicating its initial condition; for instance, A2C_1 denotes the A2C SA trained in a DT modeled under Conditions C1.

Fig. 3 shows the cumulative rewards reported by all active SAs across all the considered phases. At time $t_1 = 0$, the UAVs operate with a pre-trained model $M = 0$ until a new model is made available by the Model Evaluator. This first update occurs at $t_o = 60$ s, when the four SAs deliver the initial results of their training. From that moment onward, every 10 s the four SAs transmit their updated models together with the corresponding cumulative rewards to the Model Evaluator, which compares them against the previously received ones. As outlined in Section 2, whenever a SA exhibits better performance than the best SA in the previous period, the Model Evaluator updates the Orchestrator with the new model, which is also stored in the Model DB.

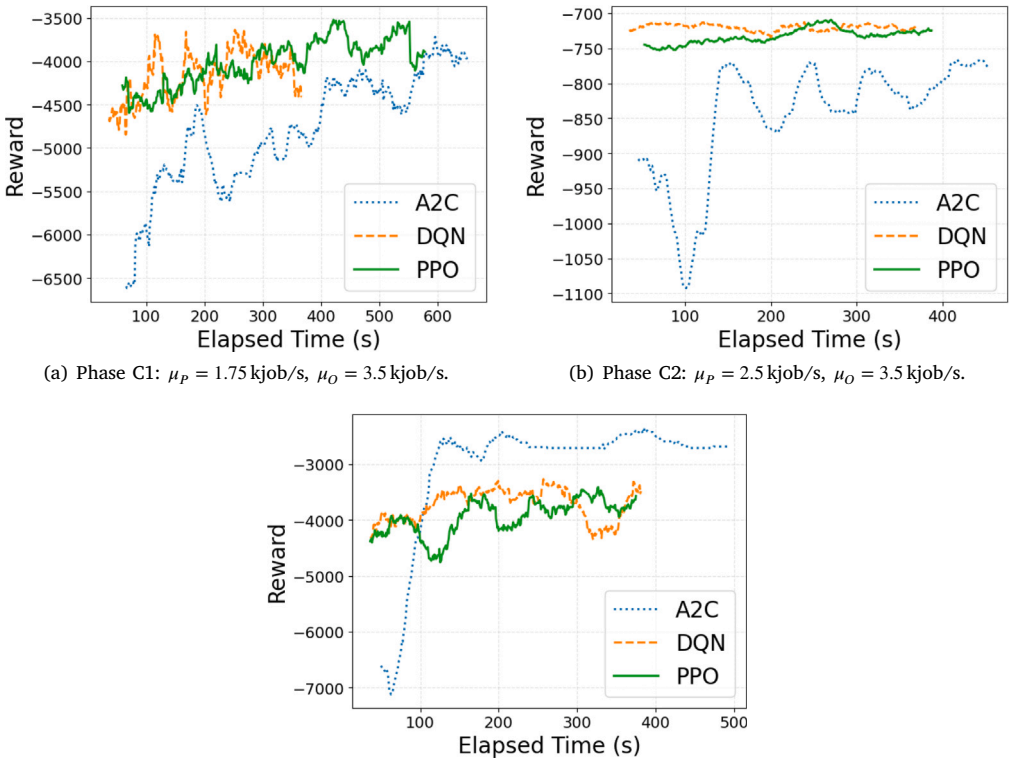
The episode rewards for the three SAs in C1 (see Fig. 3(a)) show that the DQN SA is initially the best one, closely followed by PPO. Around $t \simeq 400$ s, the PPO SA overtakes and remains the best performer for the rest of C1. In this phase, A2C never becomes the top model. Consistent with the Orchestrator's policy, the deployed model is DQN in $[t_1, 400\text{ s})$ and PPO in $[400\text{ s}, t_2)$. Training continues briefly after the C1 → C2 transition: DQN_1 stops at about $t \simeq 380$ s, PPO_1 at $t \simeq 640$ s, and A2C_1 at $t \simeq 680$ s.

At $t = t_2$, the system enters C2 (Fig. 3(b)). New SA instances are launched for C2, while the C1 instances that have not finished yet (i.e., A2C_1) keep running to completion. In C2, the best model is DQN for most of the window, with a short interval in which PPO temporarily provides a higher reward. Specifically, the deployed sequence is DQN_2 in [600, 800] s, PPO_2 in [800, 880] s, and DQN_2 again in [880, 1000] s. The C2 training runs finish shortly after the C2 → C1 boundary ($\simeq$ 1020 s for A2C_2).

When the condition returns to C1 at $t = t_3 = 1000$ s, no new training is required: the Orchestrator reloads the last best SA model (PPO_1) from the Model DB and deploys it throughout the [1000, 1400] s interval. During this period, previously launched C2 instances simply complete; no additional C1 instances are started.

At $t = t_4 = 1400$ s, the system shifts to C3 (Fig. 3(c)). Here the deployed model starts with DQN_3 but is quickly replaced by A2C_3, which achieves the highest reward for the remainder of C3. In our runs this corresponds to DQN in [1400, 1500] s and A2C in [1500, t_5] (end of C3). The associated C3 trainings end around $t \simeq 1800$ s (DQN_3), t $\simeq$ 1820 s (PPO_3), and t $\simeq$ 1880 s (A2C_3).

Fig. 5 reports the evolution of the average end-to-end delay in the FANET for the proposed framework and for three reference systems in which each learning model (A2C, DQN, PPO) operates independently, without the support of the Digital-Twin-based orchestration. The proposed solution consistently yields the lowest delay across the entire time horizon, confirming its ability to adapt rapidly to changing network conditions. During each condition, the red curve corresponding to the proposed framework remains well below that of the stand-alone learners, demonstrating that the joint orchestration of multiple SAs and the reuse of previously converged models effectively minimizes reaction time after each transition.



(a) Phase C1: $\mu_P = 1.75$ kjob/s, $\mu_O = 3.5$ kjob/s.

(b) Phase C2: $\mu_P = 2.5$ kjob/s, $\mu_O = 3.5$ kjob/s.

(c) Phase C3: $\mu_P = 1.75$ kjob/s, $\mu_O = 4.5$ kjob/s.

Fig. 3. Episode reward as the training progresses.

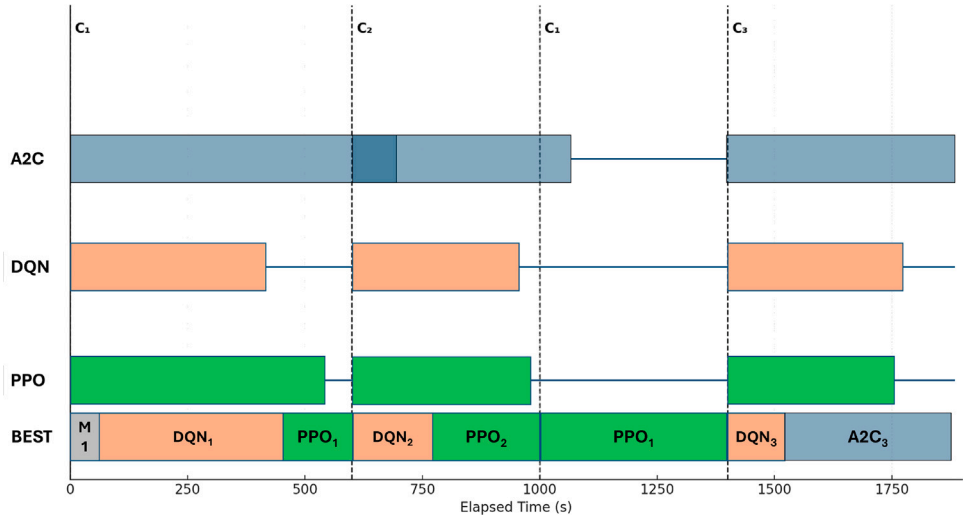


Fig. 4. Time evolution of the SA instances and models applied by the Orchestrator over time.

The A2C-based system exhibits the largest delay fluctuations, especially in the early and transition phases, largely due to its on-policy nature and slower convergence. The DQN-based system achieves a more stable but higher delay, since the discrete-action approximation struggles to generalize when continuous control adjustments are required. The PPO-based learner reacts more smoothly and reaches good steady-state values but still suffers performance drops whenever the underlying condition changes, because it lacks cross-condition memory and must re-optimize its policy for each new condition.

Conversely, the proposed DT-based orchestrated framework continuously monitors the performance of all concurrent SAs, selects the one currently providing the best reward, and reuses previously trained models when the same condition reappears. This dynamic model reuse and coordination explain why its delay remains minimal and stable even during transitions, while independent learners, trained in isolation, exhibit the typical weaknesses of their learning paradigms and fail to maintain optimal performance in all scenarios.

5. Conclusions

This article has proposed FALCON, a Digital-Twin-based framework aimed at enabling network intelligence in future 6G systems. The architecture relies on a dedicated DT of FANETs, distributed in the edge-cloud continuum. The main goal of FALCON is to interact with

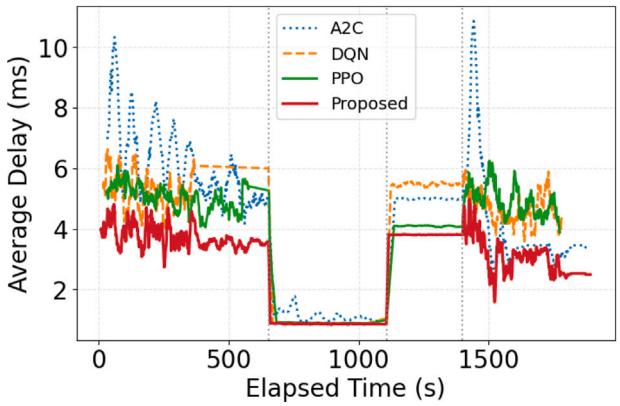


Fig. 5. Comparison of the delay achieved with the proposed framework.

the DT instances to evaluate, compare, and forecast the effects of operational decisions through parallel what-if scenario analysis. At each relevant change in system conditions, a new set of Smart Agents is instantiated to compute the model that best satisfies the intents provided by the network manager. FALCON is designed as a general and technology-agnostic platform, deployable on any FANET of UAVs capable of hosting a computation-enabled payload. Its software components can be implemented using generic container-based virtualization and orchestration infrastructures available across edge and cloud environments, enabling portability over heterogeneous UAV platforms and supporting progressive validation through large-scale emulation and hardware-in-the-loop experiments.

A proof-of-concept focused on horizontal offloading in a FANET has been presented to validate the framework. The numerical results show that coordinating multiple learning agents and selecting the best-performing model at runtime significantly improves delay, adaptability, and responsiveness compared with standalone DRL solutions. Integration of LLM-based modules to turn human intents of the FALCON operator into machine-actionable network intents can be further investigate, while Smart Agents can be extended with online, continual, and transfer learning techniques to improve adaptation and knowledge reuse across deployments. These findings confirm the benefits of integrating Digital Twins and intelligent orchestration mechanisms for next-generation UAV-based networks.

CRediT authorship contribution statement

Andrea Caruso: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Christian Grasso:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Raoul Raftopoulos:** Writing – original draft, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Giovanni Schembra:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by EU under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on “Telecommunications of the Future” (PE00000001 - program “RESTART”), the project “Programma ricerca di Ateneo UNICT 2024-26 NOVA - Network Optimization and Vulnerability Assessment” of the University of Catania, Italy, and under NextGenerationEU PRIN 2022 Prot. n. 2022MWBFEE “6GTWIN”. C. Grasso was supported by European Union (NextGeneration EU), through the MUR-PNRR project SAMOTHRACE (ECS00000022).

Data availability

Data will be made available on request.

References

[1] F. Tao, H. Zhang, A. Liu, A.Y.C. Nee, Digital twin in industry: State-of-the-art, IEEE Trans. Ind. Inform. 15 (4) (2019) 2405–2415, http://dx.doi.org/10.1109/TII.2018.2873186.

[2] A. Fuller, Z. Fan, C. Day, C. Barlow, Digital twin: Enabling technologies, challenges and open research, IEEE Access 8 (2020) 108952–108971, http://dx.doi.org/10.1109/ACCESS.2020.2998358.

[3] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, C. Jacquenet, Network Digital Twin: Concepts and Reference Architecture draft-irtf-nmrg-network-digital-twin-arch-12, Internet Engineering Task Force, 2026, https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/12/.

[4] A. Mozo, A. Karamchandani, S. Gómez-Canaval, M. Sanz, J.I. Moreno, A. Pastor, B5GEMINI: AI-driven network digital twin, Sensors 22 (11) (2022) http://dx.doi.org/10.3390/s22114106.

[5] H.X. Nguyen, K. Sun, D. To, Q.-T. Vien, T.A. Le, Digital twin for O-RAN toward 6G, IEEE Commun. Mag. 63 (3) (2025) 174–181, http://dx.doi.org/10.1109/MCOM.003.2400016.

[6] A. Caruso, C. Grasso, R. Raftopoulos, G. Schembra, IDLE: A digital twin framework for 6G network intelligence, in: 2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD, 2024, pp. 1–6, http://dx.doi.org/10.1109/CAMAD62243.2024.10942700.

[7] M. Sanz Rodrigo, D. Rivera, J.I. Moreno, M. Àlvarez-Campana, D.R. López, Digital twins for 5G networks: A modeling and deployment methodology, IEEE Access 11 (2023) 38112–38126, http://dx.doi.org/10.1109/ACCESS.2023.3267548.

[8] L.U. Khan, W. Saad, D. Niyato, Z. Han, C.S. Hong, Digital-twin-enabled 6G: Vision, architectural trends, and future directions, IEEE Commun. Mag. 60 (1) (2022) 74–80, http://dx.doi.org/10.1109/MCOM.001.21143.

[9] H.Y. Adarbah, M. Sookhak, M. Atiquzzaman, A digital twin environment for 5G vehicle-to-everything: Architecture and open issues, in: Proceedings of the Int’l ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, Association for Computing Machinery, New York, NY, USA, 2023, pp. 115–122, http://dx.doi.org/10.1145/3616394.3618266.

[10] L. Lei, G. Shen, L. Zhang, Z. Li, Toward intelligent cooperation of UAV swarms: When machine learning meets digital twin, IEEE Netw. 35 (1) (2021) 386–392, http://dx.doi.org/10.1109/MNET.011.2000388.

[11] G. Shen, L. Lei, Z. Li, S. Cai, L. Zhang, P. Cao, X. Liu, Deep reinforcement learning for flocking motion of multi-UAV systems: Learn from a digital twin, IEEE Internet Things J. 9 (13) (2022) 11141–11153, http://dx.doi.org/10.1109/JIOT.2021.3127873.

[12] F. Qi, W. Xie, L. Liu, T. Hong, F. Zhou, UAV digital twin based wireless channel modeling for 6G green IoT, Drones 7 (9) (2023) http://dx.doi.org/10.3390/drones7090562.

[13] Y. Li, D. Van Huynh, T. Do-Duy, E. Garcia-Palacios, T.Q. Duong, Unmanned aerial vehicle-aided edge networks with ultra-reliable low-latency communications: A digital twin approach, IET Signal Process. 16 (8) (2022) 897–908, http://dx.doi.org/10.1049/sil2.12128.

[14] E. Valencia, F. Toapanta, G. Oña, A. Carrillo, D. Aláez, E. Loyaga, W. Chamorro, P. Cruz, J. Abad, P. Vandewalle, An open-source UAV digital twin framework: A case study on remote sensing in the Andean mountains, J. Intell. Robot. Syst. 111 (2) (2025) http://dx.doi.org/10.1007/s10846-025-02276-7.

[15] S.K. Moorthy, A. Harindranath, M. McManus, Z. Guan, N. Mastronarde, E.S. Bentley, M. Medley, A middleware for digital twin-enabled flying network simulations using UBSim and UB-ANC, in: 2022 18th International Conference on Distributed Computing in Sensor Systems, DCOSS, 2022, pp. 322–327, http://dx.doi.org/10.1109/DCOSS54816.2022.00059.

[16] D. Cauwe, C. Wang, Q. Han, GazeboNS3: A digital twin system for UAV swarms, in: Proceedings of the 23rd Annual International Conference on Mobile Systems, Applications and Services, Association for Computing Machinery, New York, NY, USA, 2025, pp. 699–704, http://dx.doi.org/10.1145/3711875.3737659.

[17] B.M. Mohammad El-Basioni, Data-driven joint routing, topology, and mobility design for FANET systems using a digital twin approach, J. Electr. Syst. Inf. Technol. 12 (1) (2025) http://dx.doi.org/10.1186/s43067-024-00185-7.

[18] G. Faraci, C. Grasso, G. Schembra, Design of a 5G network slice extension with MEC UAVs managed with reinforcement learning, IEEE J. Sel. Areas Commun. 38 (10) (2020) 2356–2371, http://dx.doi.org/10.1109/JSAC.2020.3000416.

[19] C. Grasso, R. Raftopoulos, G. Schembra, Smart zero-touch management of uav-based edge network, IEEE Trans. Netw. Serv. Manag. 19 (4) (2022) 4350–4368, http://dx.doi.org/10.1109/TNSM.2022.3160858.

[20] G. Cappello, G. Colajanni, P. Daniele, L. Galluccio, C. Grasso, G. Schembra, L. Scrimali, Optimizing FANET lifetime for 5G softwarized network provisioning, IEEE Trans. Netw. Serv. Manag. 19 (4) (2022) 4629–4649, http://dx.doi.org/10.1109/TNSM.2022.3193883.