WIP: Ad-Hoc Network Serverless Scheduling in an Industrial Case Study: Drone Swarms in Disaster-struck Urban Environments

F. Bernardi*, S. Giallorenzo†, R. Morelli*, A. Remus*, A. Santopaolo*, F. Schiano*, A. Trotta†, G. Zavattaro†
*Leonardo S.p.A., Italy
Email: {francesco.bernardi01,roberto.morelli,alberto.remus,alessandro.santopaolo.ext,fabrizio.schiano}@leonardo.com
†Alma Mater Studiorum – Università di Bologna, Italy
Email: {saverio.giallorenzo,angelo.trotta5,gianluigi.zavattaro}@unibo.it

***Abstract*—Unmanned aerial vehicles (UAVs) are increasingly prominent in emergency management, but current approaches rely on mission-specific software stacks preloaded on each drone, which limits the swarms' flexibility when facing dynamic scenarios. Ad-Hoc Allocation Priority Policies (AHAPP) is a recent interpretation of serverless computing for ad-hoc drone networks that removes the limitation of mission-specific software stacks by allowing users to schedule and execute functions on a swarm of drones that can capture and generalise different mission scenarios. The contribution of this work-in-progress paper is twofold. On the one hand, we consolidate AHAPP by applying it for the first time to a realistic industrial case study. On the other hand, we progress towards our long-term objective, i.e., the definition of software architecture engineering practices supporting the adoption of the serverless approach in ad-hoc networks. More precisely, we show how AHAPP can capture a practical industrial case study where a multi-drone mission runs AI-based perception and coverage algorithms in disaster-struck urban environments. We start from a description of the case study architecture – including task-specific pipelines for topological distribution of UAVs and vehicle and pedestrian detection – and we interpret it in AHAPP as a system of distributed serverless functions scheduled over a mobile ad-hoc overlay. To validate our reinterpretation, we present simulations where realistic parameters from the case study – including detection frame rates, heterogeneous compute loads, and a motion-versus-compute energy breakdown – drive experiments that scale from three to thirty drones and from a few city blocks to larger areas. The results show that the scheduling modes of the proposed interpretation afford high deployment success, acceptable delay, and bounded overhead when applied to industrial-grade constraints and workloads.**

***Index Terms*—Serverless computing, ad-hoc networks, UAV swarms.**

I. Introduction

Unmanned aerial vehicles (UAVs) swarms are becoming a reliable technology for time-critical applications such as disaster-aware search-and-rescue missions where, e.g., aerial views help maintain situational awareness when ground infrastructure is partially unavailable or overloaded. Typical state-of-the-art industrial applications see multiple drones coordinate to e.g., patrol an urban area, detect vehicles and pedestrians, and feed detections to a central management component that aggregates patterns of interest (e.g., hazardous, congested). However, these systems typically assume an imperative deployment model, in which designers pre-program missions and preload the functionality needed on each drone, limiting the ability to introduce new capabilities or adapt to unforeseen conditions without returning drones to base for reconfiguration.

Serverless Function-as-a-Service [1] (FaaS) is a Cloud architectural pattern that allows users to deploy and schedule code execution, wrapped within a function, onto a pool of execution nodes. Ad-Hoc Allocation Priority Policies [2] (AHAPP) implements a recent interpretation of edge-cloud continuum FaaS [3] for ad-hoc drone networks that supports the decentralised scheduling of functions on nodes that may differ in connectivity, compute, energy, and sensor equipment. The ability to schedule functions on drones removes the limitation of mission-specific software stacks by allowing users to schedule and execute on a swarm of drones code that implements different mission scenarios.

In this work-in-progress paper, we consolidate such ad-hoc serverless architecture by showing that it can host and extend an industrial disaster-aware monitoring case study. We consider this contribution a relevant step forward in the definition of novel software architecture engineering practices supporting the adoption of the serverless approach in ad-hoc networks.

II. Industrial Case Study

The case study targets disaster-aware traffic monitoring over an urban area, using a swarm of drones equipped with simulated flight controllers, environmental sensors, and cameras. The control architecture combines ROS 2 [4] for modular and distributed control, PX4 SITL for low-level flight dynamics, DDS middleware for secure and QoS-aware communication, and the AirSim simulator [5] for high-fidelity environments and sensor models. Within this environment, a swarm of drones takes off, divides a disaster-struck city sector, patrols assigned subregions, and runs multi-level image recognition routines. Figure 1 illustrates the main steps of the described case.

In particular, the drone coverage uses a grid-based representation of the monitored area, followed by a DARP-style [6] partitioning that assigns disjoint subregions to each UAV, and a spanning-tree coverage algorithm that generates energy-efficient paths. On top of this control stack, the UAVs host trained and integrated object detectors for vehicles and

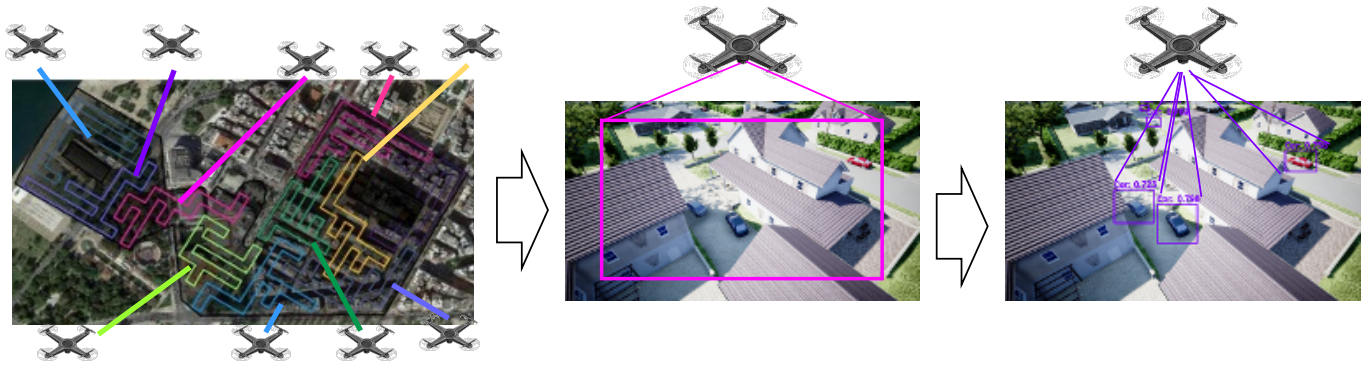

Fig. 1: Depiction of the disaster-aware case study.

```
targets: all
attributes:
 position: { lat: LAT, lon: LON,
             alt: ALT, range: RNG }
 device:
    camera: { lock: true, quality: HD },
    gpu: { lock: true }
```

Fig. 2: Example AHAPP scheduling policy.

pedestrians using a dataset derived from AirSim, performing hyperparameter optimisation and model comparison across variants of the same model to balance detection accuracy and inference latency under drone-specific constraints. Among these, we select a "default" version that works at around 3 frames per second (fps) in "low-load" mode to monitor a region of interest, while an optimised version achieves approximately 30 fps for full detection and tracking of traffic.

As part of the case study, hardware-in-the-loop experiments with a Jetson Orin board that acts as the onboard computer revealed that propulsion accounts for roughly 90% of the drone's total energy, while onboard compute contributes about 10%. Despite propulsion consumes most of the energy, since long range missions require saving as much battery as possible, it is important to manage power consumption due to onboard computing. For the perception workload, the Share Region-of-Interest (ROI) task corresponds to a low-CPU execution regime (approximately 10% of the 25 W peak, i.e., a few watts over short time windows), whereas the optimised detection version uses most of the GPU budget and approaches the 20 W range of the Jetson module. These findings motivate an architecture where compute-intensive functions such as high-rate detection and tracking are explicitly scheduled in a way that is aware of local energy and compute budgets.

III. Ad-Hoc Serverless Framework

Before modelling the case, we briefly introduce the essential elements of the AHAPP framework used in the modelling. The interested reader can find the details about all the elements of the framework in its seminal presentation [2]. AHAPP defines a two-layer overlay on top of the physical topology of drones and base stations. A gossiping layer continuously disseminates compact summaries of node status, including neighbourhood information, available memory and storage, residual energy, sensor and accelerator availability (e.g., cameras and GPUs), and an approximate description of the evolving connectivity graph. To execute functions, users send to a base station code to be run on drones, relying on a scheduling layer to route and deploy the functions. Technically, users issue function-execution requests coupled with a scheduling policy that defines which drones can execute that function. The function-scheduling layer supports the decentralised routing of the request from the base station, through the network of drones, to the function's suitable executors.

In this paper, we consider a *hierarchical* scheduling logic, where the base station, given the information from the gossiping level, identifies which drones should run a given function and the drones mainly act as routers and executors.

Users specify scheduling policies using the declarative language AHAPP (which gives the name to the framework) to define scheduling requirements and preferences together with each function. Policies can constrain sensor and hardware features (e.g., presence of a GPU-accelerated detector, camera availability), physical position (e.g., inside a specified region), resource thresholds (e.g., minimal residual energy and memory), and they can express a selection strategy that prioritises targets according to attributes like distance and energy.

To illustrate AHAPP's grammar, we present a policy modelling the scheduling of a function that must run on all drones within a given area, equipped with both camera and GPU. Figure 2 shows an AHAPP policy that captures the above scheduling requirements by configuring the `targets` option to select `all` nodes that satisfy the `attributes` block,

which constrains the eligible ones based on their `position` and hardware `device` capabilities, such as `camera` and `gpu`, which must be available for `lock`ing, so that the function under scheduling can get their exclusive use.

IV. Mapping the Case Study to Serverless

We reinterpret the workflow of the case study as a set of serverless functions deployed over the ad-hoc overlay, instead of a mission-specific control stack. At a high level, the scenario includes (i) coverage management functions, whereby `gridDiscretisation` implements grid discretisation and partitioning and `coveragePlanning` determines a drone hovering pattern, and (ii) perception functions, whereby `ROI-detection` implements a Share Region-of-Interest logic and `HR-detection` performs fine-grained object detection. Each function is associated with an AHAPP policy (see Listings 1–4) that describes its required sensors and compute accelerators, drone positions, and minimum residual energy.

The `coveragePlanning` function targets drones that run the `gridDiscretisation`, i.e., they have sufficient energy to complete the mission workflow. These policies emphasise positional constraints and residual battery. Perception functions come in two profiles: a low-rate `ROI-detection` that runs at about 3 fps and assumes modest CPU usage, and a high-rate detector (`HR-detection`) that sustains around 30 fps but saturates GPU resources. The corresponding policies for the high-rate detector require exclusive access to a GPU.

In AHAPP's paradigm, the user expresses execution objectives – for example, "scan all patrolled areas for congestion" or "increase detection density around a suspected bottleneck" – as serverless function-execution requests that the platform translates into function deployments according to network conditions. Drones that happen to be in the right area with sufficient (and unlocked) resources can execute the requested functions, while others may act as forwarders, exploiting the ad-hoc overlay to route both code and results.

V. Simulation Methodology and Results

We evaluate the feasibility and scalability of the proposed AHAPP-based reinterpretation of the case study (cf. Section II) through discrete-event simulations carried out in OMNeT++ [7] using the INET framework. The simulator extends the implementation presented in AHAPP's seminal work [2] by incorporating realistic parameters derived from hardware-in-the-loop experiments (cf. Section II), including heterogeneous perception workloads, mobility constraints, and energy-aware scheduling policies.

We consider a disaster-aware monitoring scenario where a swarm of UAVs patrols an urban area while running a low-resolution perception function (`ROI-detection`) on all drones. At runtime, a base station triggers the deployment of a high-resolution detection service (`HR-detection`) over three randomly selected hot spots within the monitored area. Each hot spot is associated with a spatial compatibility range, and only drones that are located within such range, equipped with a camera and GPU, and above a minimum residual energy threshold are considered eligible for deployment. All requests are propagated through the ad-hoc network toward eligible drones using AHAPP's hierarchical scheduling mode. The number of drones varies from 5 to 30, and we consider three mobility situations with average drone speeds of 0 m/s (static), 10 m/s, and 20 m/s. Drones communicate over an IEEE 802.11 ad-hoc network and exchange periodic gossip messages to disseminate local status information. All results are obtained by averaging multiple independent simulation runs, and are reported with confidence intervals.

Figure 3 reports the failure ratio of `HR-detection` deployments, defined as the fraction of eligible drones that do not successfully deploy the requested service before a timeout. The failure ratio increases with both the number of drones and their mobility. Larger swarms induce longer multi-hop paths and higher contention, while increased mobility exacerbates route instability and the effects of stale network knowledge during the deployment phase. Despite these effects, the system maintains acceptable deployment success for moderate swarm sizes and under increasing mobility.

Figure 4 shows the total number of drones that successfully deploy the `HR-detection` function. Although there are higher failure ratios as the swarm grows, the absolute number of deployed services increases in the stable scenario, reflecting higher spatial density and a larger pool of eligible candidates, while it stalls with faster drones. Hence, we have improved sensing coverage and higher resolution around the incident area in stable configurations, while in larger swarms and higher mobility levels, the successful deployments decrease, as multi-hop dissemination overhead and transient connectivity prevent a significant fraction of candidates from completing the deployment. This result highlights an important trade-off between swarm densification and network scalability that needs to be tackled in mobile ad-hoc serverless systems.

Figure 5 reports the average deployment delay, computed only over successfully deployed HR services. For small swarms, deployment delays are well below one second, consistent with one-hop deployments. As the number of drones increases, delays grow due to longer paths and higher contention. Interestingly, higher mobility does not necessarily lead to higher observed delays. At higher speeds, many multi-hop deployment attempts fail, and the measured delay is dominated by the subset of deployments that reach nearby executors over short paths, resulting in bounded average latency.

VI. Discussion and Future Work

Unifying the views provided by Figures 3 to 5, we observe that AHAPP can support the coordination of the execution of realistic missions over small-to-mid-sized ad-hoc networks of drones. We remind that, in our simulations, we use a hierarchical scheduling logic that sees the base station take scheduling decisions given its knowledge of the drones obtained through gossip messages. While this modality works well on larger and larger networks, it does so as long as the drones assume relatively stable motion patterns – they mainly hover above a designated overview point. On the contrary,

Listing 1: AHAPP policy for `gridDiscretisation`.

```
targets: all
attributes:
  energy:
    min: MIN_NRG
  device:
    camera: { quality: HD }
```

Listing 2: AHAPP policy for `coveragePlanning`.

```
targets: all
attributes:
 affinity: gridDiscretisation
 position: {
   lat: LAT, lon: LON,
   alt: ALT, range: RNG }
 device:
  rotors: { lock: true }
```

Listing 3: AHAPP policy for `ROI-detection`.

```
targets: all
attributes:
 device:
   camera:
    quality: HD
```

Listing 4: AHAPP policy for `HR-detection`.

```
targets: all
attributes:
 energy: { min: 80 }
 position: {
   lat: LAT_HOT, lon: LON_HOT,
   alt: ALT_HOT, range: RNG_HOT }
 device:
   camera: { lock: true, quality: HD }
   gpu: { lock: true }
```

the faster the drones move, the quicker the information that the base station has becomes stale, driving up the number of failed deployments – e.g., a function cannot reach a designated drone following a hierarchically-established route because the topology of connections changed or it reaches the designated drone, but the drone does not satisfy the execution requirements, e.g., it is outside the position range.

These experiments motivate the investigation of other scheduling modalities able to cope with highly-dynamic scenarios where the base station cannot trust having up-to-date information about the network. This evidence validates the reasoning behind the proposal by De Palma et al. [2] to consider an alternative scheduling logic where the base station and the drones participate in the progressive definition of the execution targets of a function. In that modality, the base station does not designate the executors of a function and rather forwards the execution request to nearby drones that are in the direction of targets that might be able to run it. In turn, the intervening drones decide which targets could run the function and proceed to forward the message (and possibly run the function themselves).

While De Palma et al. [2] define these two modalities as dual, the experience we gathered from our experiments leads us to conjecture that a hybrid mode could be better suited to adapt to changing system dynamics and the natural core-periphery segmentation of the ad-hoc network. In this modality, the base station (at the network's core) would take hierarchical scheduling decisions but the intervening nodes would compare the decision taken by the base station with their current knowledge of the network; if the scheduling decision taken by the base station becomes invalid when compared to the (more up-to-date) information of the drone, the drone can switch the scheduling mode of that function to the progressive one and proceed to find a suitable target in that way. We believe that this hybrid modality could effectively address the limitations of either imposing a hierarchical or progressive modality and adapt both to large and highly dynamic networks.

This work has been partially supported by PNRR CN HPC - SPOKE 9 - Innovation Grant LEONARDO - TASI - RTMER funded by the NextGenerationEU European initiative through the MUR, Italy (CUP: J33C22001170001) and by French ANR project SmartCloud ANR-23-CE25-0012.

References

[1] E. Jonas *et al.*, "Cloud programming simplified: A berkeley view on serverless computing," *CoRR*, vol. abs/1902.03383, 2019.
[2] G. De Palma *et al.*, "Distributed serverless function scheduling in ad-hoc drone networks," *Ad Hoc Networks*, vol. 178, p. 103951, 2025.
[3] M. S. Aslanpour *et al.*, "Serverless edge computing: Vision and challenges," in *ACSW '21: 2021 Australasian Computer Science Week Multiconference, Dunedin, New Zealand, 1-5 February, 2021*. ACM, 2021, pp. 10:1–10:10.
[4] S. Macenski *et al.*, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robotics*, vol. 7, no. 66, 2022.
[5] S. Shah *et al.*, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *FSR 2017*, ser. Springer Proceedings in Advanced Robotics, vol. 5. Springer, 2017, pp. 621–635.
[6] A. C. Kapoutsis *et al.*, "DARP: divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robotic Syst.*, vol. 86, no. 3-4, pp. 663–680, 2017.
[7] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, SimuTools 2008, Marseille, France, March 3-7, 2008*. ICST/ACM, 2008, p. 60.

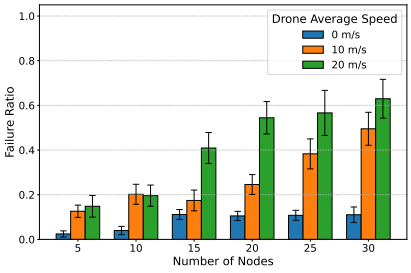

Fig. 3: Failure ratio of `HR-detection` deployments vs # nodes and velocity.

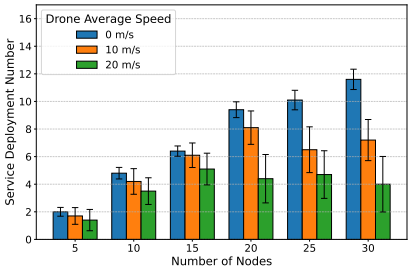

Fig. 4: Total number of successfully deployed `HR-detection` function.

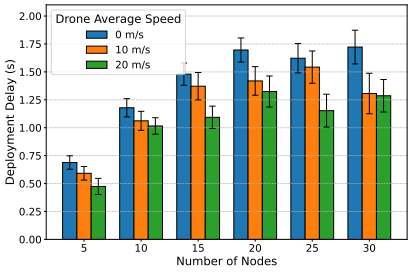

Fig. 5: Avg. delay of successful `HR-detection` deployments.