The AERPAW Control Framework - Considerations for Resource Control and Orchestration for a Computing-supported Physical Research Platform

Magreth Mushi[a], Harshvardhan P. Joshi[a], Rudra Dutta[a],
Ismail Guvenc[b], Mihail L. Sichitiu[b], Thomas Zajkowski[c]
[a]*Dept. of Computer Science*,
[b]*Dept. of Electrical and Computer Engineering*,
[c]*Institute for Transportation Research and Education, Aviation*
North Carolina State University, Raleigh, NC, USA
{mjmushi, hpjoshi, rdutta, iguvenc, mlsichit, tjzajkow}@ncsu.edu

Yufeng Xin, Michael J. Stealey, Erik L. Scott
Renaissance Computing Institute (RENCI),,
University of North Carolina - Chapel Hill
Chapel Hill, NC, USA
yxin@renci.org, stealey@unc.edu, escott@renci.org

***Abstract*—The AERPAW project is an ambitious project, funded by the PAWR program of the US NSF, to create a remote accessible research platform for a research facility with some distinct features that makes its operational model unique, and possibly non-obvious to computing researchers more familiar with computing and networking testbeds. AERPAW is primarily a platform of physical resources - specifically the RF environment, and the airspace. Experimenters can explore them through radio transceivers and Unmanned Aerial Vehicles, both under the Experimenter's programmatic control. The entire workflow of the remote Experimenter is through the mediation of virtual computing environments, but AERPAW Operators have to ensure that the Experimenter's programming controls the physical resources precisely as intended by the Experimenter, while also satisfying safety and compliance goals. To this end, the AERPAW platform adopts multiple distinct, and quite different, execution environments. This in turn gives something of a different color to the well-understood problems of resource allocation, scheduling, and access control. In this paper, we articulate lessons we have learned, and the challenges and considerations we have come to appreciate, in designing a resource control framework for such a physico-cyber facility.**

***Index Terms*—Wireless, testbed, 5G, NextG, UAV, UAS, drone**

I. Introduction

The wireless era is new in the history of the modern world, with unprecedented bandwidths and ubiquitous mobility unleashing new applications and capabilities. Revolution of so many things connected wirelessly in a way never done before allows technologies such as driverless cars, virtual/augmented reality, drones, and massive Internet of things (IoT) moving into reality. The fifth generation (5G) cellular networks will offer unprecedented data rates, ultra-reliable low-latency communications, and massive machine type connectivity that will allow enterprises and specific use cases to flourish in ways we now cannot predict. Central to the vision of 5G is the availability of immense bandwidths that will enable, for the first time ever, cognition and video control of robots, and moving platforms, particularly in areas where the dense 5G wireless network exists.

ISBN 978-3-903176-48-5

Major new use cases for advanced wireless technologies, once thought to be science fiction, are emerging in the unmanned aerial systems (UAS) spaces [1]–[3]. Major package delivery companies such as Amazon and UPS [4], [5] have invested heavily in UAS capabilities, and many other military, government, and consumer applications of UAS have attracted major attention, e.g. by Google [6], Uber [7], Boeing [8], Zipline [9], Flytrex [10], and Matternet [11], among others. The U.S. wireless industry will provide the infrastructure and capabilities to service this revolutionary move to UAS and flying vehicles. While various cellular providers (AT&T [12], Vodafone [13]), vendors (Ericsson [14]), and chip manufacturers (Qualcomm [15]) have recently been studying cellular connectivity for aerial links, yet very little is known to date about how to properly design, develop, analyze, and test this major leap in the world's infrastructure of UAS.

The Platforms for Advanced Wireless Research (PAWR) funding program of the US National Science Foundation (NSF), with funding provided both by NSF and an industry consortium engaging public and private partners, has the goal of establishing several large scale advanced wireless testbeds enabling experimental research at scale. The PAWR-funded testbeds will be open to both academic and industry researchers, enabling unprecedented access to advanced wireless technologies. Participating companies provide in-kind investment and receive in return priority use of the corresponding testbeds, potentially presenting research experimentation opportunities not available elsewhere.

The Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) is the third such testbed funded under the PAWR initiative. The project started in September, 2019, and the AERPAW facility became generally available, with a minimal set of experimental resources, for the first time in November, 2021. Over the next two years, the set of experimental resources is expected to increase.

The unique nature of AERPAW poses both an opportunity and a challenge in terms of providing the full value of the facility to the researcher community intending to use it for the most ambitious experiments. On the one hand, we wish to provide the researcher with the unfettered capability of

studying whatever experimental scenarios their research calls for; on the other, we not only need to guarantee compliance with relevant regulations, but also ensure safety of operating personnel (and any bystanders), as well as testbed equipment and surrounding property. As an example, AERPAW UASs (custom-designed and built to maximize user-programmability and optimize radio and ohter payloads) are 30 Kg (without payload) hexacopters, with 23-inch carbon-fiber propellers. They can cause serious injury or very possibly death in the case of a worst-case impact with a human if operated incorrectly. This motivates us, even as we hand over complete programmatic controllability to a researcher, the need for oversight and, if necessary, override during operation.

II. AERPAW Facility

AERPAW was envisioned to fill the need of enabling meso-scale UAS experiments in a production-like wireless environment, and serve as both an advanced wireless research facility, and a UAS research facility; but most importantly, provide a testbed for research that jointly addresses both.

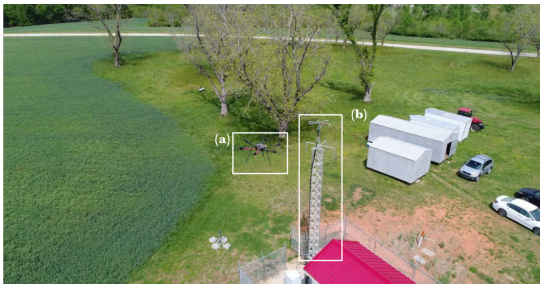

Fig. 1: AERPAW experiment in progress at Lake Wheeler: (a) one UAV mounted portable node and (b) one tower-mounted fixed node is visible

Physically, the testbed is hosted at sites in and around the NC State University campus in Raleigh, NC, USA. Central to AERPAW's unique characteristic is the availability of Unmanned Aerial Vehicles (UAVs) in the testbed that can be placed under the direct programmatic control (of trajectories) of the researcher. In conjunction with the programmable USRPs also available for direct programming by the researchers, as well as other real-world, commercial radio equipment, this provides the NextG wireless researcher a facility for research experiments not practicable in any other facility at this time.

At a very high level, the facility includes a number of *Tower* locations, at each of which some combination of AERPAW programmable SDRs and commercial radio equipment is permanently installed. The SDRs are controlled by servers, or companion computers, installed in each location that represent edge-computing capabilities. These fixed node locations are distributed over the extensive Lake Wheeler Agricultural Fields of the NC State University, and some nodes are also installed in the more conventional Centennial campus of NC State. In future, some nodes may be installed in adjoining city and town areas. The complement of these Fixed Nodes are AERPAW's Portable Nodes, also consisting of a computer and SDRs, but smaller ones so that an AERPAW Portable Node can be mounted on a UAV. The computer on a Portable Node also controls the UAV itself.

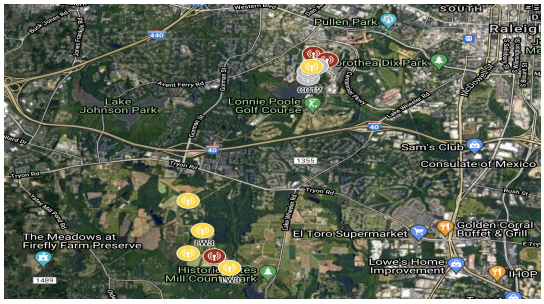

Fig. 2: AERPAW initial deployment area

To provide an overall idea of the testbed expanse, Fig. 2 shows the location of currently active (red) fixed nodes as well as those to be commissioned in the near future (yellow). To provide a sense of scale, the north-south extent of the area shown is just over three-and-half miles. The bottom left cluster of yellow nodes occupy a central position in the Mid-Pines Lake Wheeler Field Laboratory of NC State, where further buildout is planned for future.

To be able to conduct such experiments, while remaining in compliance with the stringent regulations of the Federal Communications Commission (FCC) and the Federal Aviation Authority (FAA) for SDR and UAV operations respectively, requires appropriate certifications and exemptions. The AERPAW platform has already acquired the relevant ones, and extending to further exemptions is on our roadmap; these certifications and the experimental capabilities unleashed by them (and the accompanying practical experience) are critical resources that AERPAW brings to its experimenters, beyond the physical resources. The details of the certifications and exemptions are available in the AERPAW User Manual [16].

Extensive information about the facility, equipment details, software support, etc. are provided in our public online User Manual [16], the AERPAW website [17], and representative research experiments as well as design details of the platform have been described in our prior publications such as [18]–[21]. We do not repeat any of that content here, beyond the above preliminaries. Instead, we next describe some aspects of AERPAW architecture, to facilitate understanding of the requirements of experiment oversight that follows.

III. Usage Model – Design Goals and Constraints

As is clear from the above, **AERPAW is primarily and essentially a testbed of physical resources, not computing resources.** The crucial part of these physical resources are (i) the RF environment and the airspace that the AERPAW operating areas represent, (ii) the physical equipment (SDRs and UAVs) that AERPAW provides to leverage those environments for experimental studies, and (iii) the expertise (and consequent exemptions) in conducting such studies in compliance with FCC and FAA regulations that AERPAW represents.

A. Deep Access

As an NSF-funded testbed, AERPAW has the goal, and the charge, to serve a broad range of researchers nationwide, rather than just a small set of specific domain researchers with deep-but-narrow portfolio of domain expertise and interest. In other words, we do not aspire only to be of use to the few researchers, typically in industry, who are already well-versed in SDR, 4G/5G and UAS programming, and are simply seeking a physical arena for their scale experiments. Rather, we also aspire to bring the capability of engaging in such research to a larger variety of researchers who may not already possess such expertise, including those in academia, with a broader range of interests – e.g. in the theory and analysis of channel models or modulation schemes, in algorithmic trajectory control and optimization, in edge-computing for real-time image analysis or cyberphysical systems, and many others.

Further, we wish to provide researchers with complete control of the equipment we operationalize in the testbed, without imposing any additional artificial barriers, or limitations of planned experimental activity. In other words, we aim to provide the researcher with "root access" to all our equipment. This is not to say that arbitrary experimental activity can be *executed* on the testbed without limitation; naturally, unsafe operation (such as RF operation outside the allowed profile of frequency and transmission strength, or UAV operation outside the allowed profile of speed, angle of attack, etc.) must be prevented from actually occurring. We describe this more specifically below, and relevant discussion may also be found in [18]. Rather, we make the choice to try to preclude such unsafe or illegal operation at actual execution time, rather than by imposing *a priori* limitations on the experiment design and planning process, or tools used for that purpose (to ensure that no "dangerous" experiments can even be defined). The reason for doing so is that the latter choice, though easier to implement for AERPAW, would take the nature of blanket bans, and preclude many interesting and legal experiments as well as actually illegal ones. For example, consider the researcher who writes a custom algorithm for dynamically selecting the transmission frequency or power, or UAV trajecory (or both) during execution. Depending on the nature of the algorithm, it might well be impossible to predict with certainty (even for the researcher himself/herself) whether the operation will stay within legal bounds during actual execution. The only way to preclude dangerous operation with certainty would be to disable programmable real-time computed control of the SDRs or UAVs altogether (and insist instead on explicitly predetermined and pre-programmed transmission and flight), thus precluding many interesting algorithmic experiments. Finally, in future, we aspire to provide researchers with similar deep control, within practical feasibility, of real-world commercial radio equipment (such as commercial cellular base stations).

B. Regulatory Compliance Issues

Since the AERPAW testbed is physically located in the particular space it occupies, to physically access it researchers would have to travel to it, which is impractical. Therefore we must make it possible to use this testbed remotely. This, and the goal of providing full control to the researcher, requires that every capability of AERPAW equipment must be possible to access by the mediation of some computing device – most typically, commodity computers running a general purpose OS (usually Linux), but also special-purpose software to enable control of radio or UAV equipment. The remote experimenter can optionally choose to write their own custom software extensions or replacements for any part of the pre-installed software.

This, incidentally, may potentially reinforce the *incorrect* impression that AERPAW is fundamentally a computing, or cloud-computing, facility, with some peripherals accessible through the remotely accessible computers. However, as we have described above, the appropriate way to think of AERPAW is as a physical facility, mediated by computing for the dual purposes of remote access, and deep programmability.

The second issue raised by this set of goals and constraints is that of liability. AERPAW possesses appropriate exemptions and permits, as described above, to conduct a range of RF and UAV operations, and can execute such experiments legally, these permits do not automatically translate to allowing other researchers to conduct the same operations. Thus, a researcher who desires to perform an experiment on AERPAW needs to first obtain the same exemptions, certifications, and permits as AERPAW, which is obviously untenable in terms of using AERPAW as a testbed. It would also not be practical for AERPAW to verify such permits. Further, if unsafe operation occurred, it would be unclear where the liability should lie – with AERPAW, or the experimenter.

C. Batch-mode Access

To reflect the main purpose outlined above, balancing the needs to afford deep programmability to the researcher using AERPAW, and to ensure safety and regulatory compliance, we reached the conclusion that the canonical usage model of AERPAW must be that of *batch-mode access*. In the computing context, this feels like an archaic model; however, it is a standard model for facilities that are fundamentally physical in nature, such as large telescopes or microscopes. Batch-mode operation is thus a natural fit for an essentially physical facility such as AERPAW.

However, the AERPAW canonical model is somewhat more ambitious than the standard batch-processing model for large physical resources. Most such models not only allow but actually require the user to not only produce written documents to describe the intended experimental activity, but also to write executable scripts that actually drive the physical device(s). The same is true of AERPAW. However, our goal go significantly further, in two ways. First, for canonical experiments, we hope to eliminate the step of documentary or any other human-in-the-loop interaction altogether, and allow the AERPAW experimenter to define an experiment and stage it for batch execution completely programmatically. At the time of this writing, some human-in-the-loop operation is still

required, although transparent to the user; but we expect to eliminate it soon with ongoing development of the platform. Note that custom experiments that do not fit the automated workflow can always be imagined, and such experiments will always require human-mediated planning; see below for more details. Also note that even for canonical experiments, human action is required for the actual execution of the experiment, since UAVs have to be staged, tested for safety, etc.; what is automated is the process of experiment planning and batch submission; again see below. Secondly, we aim to provide the researcher with a single experiment preparation methodology that will enable them to write the minimal scripts for using AERPAW for oft-repeated experiments, but also to use the same environment to develop sophisticated custom software extensions of their own. For example, a researcher may want to encode their own experimental RF waveform for specialized use such as air-to-ground communications.

D. Staged Preparation Discipline

To enable this vision, a complementary virtual and physical access discipline was envisioned by the AERPAW architects. During experiment preparation, researchers using AERPAW are provided access (with root permissions) to a number of virtual computing nodes, one corresponding to each physical AERPAW computing node. From the user's vantage point, each node is identical to the corresponding testbed node; in fact, they are identical in software capabilities, but do not have physical resources such as SDRs or UAVs (or other commercial radio equipment) connected to them. However, the software behaves the same general way in this environment as in the actual testbed environment, thanks to a custom emulation environment, described more fully in [18]. This not only enables the experimenter to develop their software with ease, running it in the emulated environment through the development cycle, but also makes for a seamless transfer of that software to the testbed nodes when the time for batch-mode execution comes. We call this the **Development Mode** of execution. A closely related execution mode is called **Emulation Mode**, which is identical to the Development Mode, except that the Experimenter does not have access to the virtual computing nodes in this mode (which is intended to allow the AERPAW Ops personnel to verify certain details of experiment execution and conformance before attempting Testbed execution of the experiment. The custom virtual computing environment crafted by AERPAW to support both these two execution modes is called the AERPAW **Virtual Environment** (VE). The VE resources are exclusively commodity computing and swtiching equipment, together with commodity and custom software environments.

In contrast, the AERPAW **Physical Environment** contains two classes of resources. Both contain actual AERPAW computing nodes, which have actual SDRs and hardware vehicle controllers connected to them. However, in the **Sandbox** environment, these nodes are placed in a controlled lab environment, and their behavior is curtailed. The vehicles do not actually have the capability of moving, because they are fixed in place. The SDRs are supplied with attenuating cables in lieu of antennas, so they exchange RF signals through a crafted intermediate hardware RF network, rather than a real-world RF environment. In contrast, the **Testbed** environment consists of a number of such nodes embedded in the real-world footprint of the platform (as shown in Figures 1 and 2), "in the wild", as it were. The Sandbox mode of execution is intended to allow Experimenters to obtain hands-on familiarity with the driver software for the SDR and vehicle hardware, and as such requires to support remote live access to the Sandbox resources. Conversely, the Testbed mode of execution is the final target mode of execution of all Experiments, since only this mode allows execution and data-gathering in an actual wireless and aerial field of real-world significance. In Testbed mode execution, the Experimenter does not have live remote login access to the nodes on which the Experiment executes; the Experiment is ported from the Developemnt mode to the Testbed mode by AERPAW Ops, and execution is kicked off there by an automated system called AERPAW Operator's Experiment Oversight system, discussed more fully in [21].

Fig. 3 shows the entity relationships of the platform at a very high level, that captures the above.

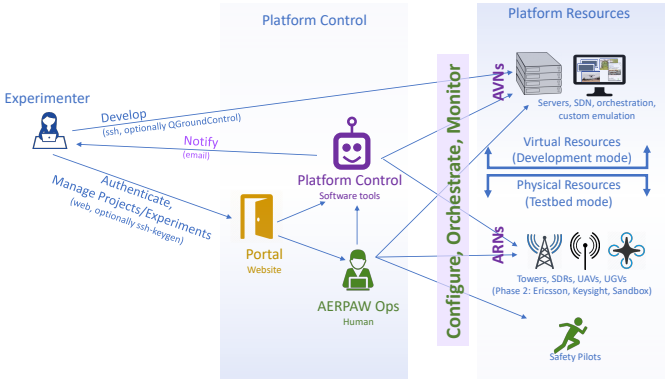

Fig. 3: AERPAW Entity Relationships

In summary, then, **AERPAW is a batch-mode facility**. Experimenters develop experiments in a virtual computing environment, and submit experiments for execution on the physical testbed once development is complete. AERPAW Operations personnel (Ops) then execute these submitted experiments in the physical testbed environment, and collect the output of the experiments as designed by the Experimenters, which are available for Experimenters to view and analyze back in the virtual environment.

This is not an arbitrarily decided constraint, but a considered architectural choice as we have discussed above. In operating a facility with programmable radios and programmable air vehicles, we are obligated to make, and uphold, certain guarantees to the FCC and FAA. However, we also want to allow Experimenters the ability to program those radios and air vehicles, ideally without needing to become fully conversant with FCC and FAA regulation details, obtaining exemptions, or expertise at techniques for ensure compliance.

Batch mode operation allows us to interpose critical filters and monitors into the Experiment code execution flow that

allow us to guarantee safe and compliant operation. It is one of the most valuable features of the AERPAW platform that we assume this guarantee ourselves, rather than passing on the responsibility for compliant operations (and liability for non-compliance) to the Experimenter.

IV. AERPAW Architectural Approach

While this paper focuses specifically on the Operator's point-of-view, and the introduction of automation to enable necessary override mechanisms, some nomenclature and concepts related to the general AERPAW architecture are required as preliminaries, which we present in this section.

A. AERPAW Node Classes

AERPAW Compute Node: A computer, real/virtual, that contains client-side AERPAW control software. We often refer to this as a Companion Computer, since it accompanies whatever experimental resource (radios, vehicles) an experimenter is interested in experimenting. We also sometimes refer to this simply as an **AERPAW Node**. An AERPAW Node that is instantiated physically for an experiment is an AERPAW Hardware Node (AHN); a VM or container that is instantiated to be a close digital twin of an AERPAW Hardware Node, for the purpose of experiment preparation, is an AERPAW Virtual Node (AVN).

AERPAW Cloud Node (ACN): This is a special case of an AERPAW Node to which no other controllable hardware resources are connected, and therefore for which the physical location is not important. For the experimenter, it represents a pure computational resource. Such a node would be instantiated as a virtual computing entity (VM or container) in the AERPAW central datacenter (DCS).

AERPAW Hardware Radio Node (AHRN): An AHN to which one or more experimental wireless hardware resources are connected. An AERPAW Radio Node implies specific physical instantiation at some location at which the radio resources exist, and is not a virtual resource (although it may contain multiple VMs or other execution contexts within itself). We distinguish between **Fixed** and **Portable**, called **AFHRN** and **APHRN** respectively. Each of these two types of AERPAW Radio Nodes can, of course, have a VM instantiation (for development or emulation), in which the experimental wireless hardware is replaced with an emulation, thus providing AERPAW Fixed Virtual Radio Nodes (AFVRNs) and AERPAW Portable Virtual Radio Nodes (APVRNs).

A Fixed Radio Node (AFHRN) is installed at some specific location, (semi-)permanently, and can only be used in-situ by experimenters. Essentially these nodes are "always-there" or "persistent." They may be powered down for intervals by testbed control but are "always-there" to be remotely powered back on.

A Portable Radio Node (APHRN) is compact and self-powered, so that it may be moved during an experimental session. As such, it is only guaranteed to be available to testbed control at specific times, thus "ephemeral." It may appear at various times with various L2 and L3 addresses when it does, but needs to be recognized by testbed control as the "same node."

Finally, some Fixed and Portable Hardware Nodes may be placed in the controlled Sandbox facility where the connected experimental wireless hardware has been modified to allow dense indoor placement. This creates AERPAW Sandbox Fixed Hardware Radio Nodes (ASFHRNs) and AERPAW Sandbox Portable Hardware Radio Nodes (ASPHRNs).

B. Entities Associated with AERPAW Nodes

AERPAW Experimental Radio: Any wireless equipment to which AERPAW will provide experimenter access through a Companion Computer. A large fraction of these are expected to be SDRs, but other commodity wireless equipment such as WiFi or LoRaWAN also fall into this class. The equipment we consider in this class have open interfaces, and are deeply programmable or at least comprehensively configurable. An AERPAW Radio Node houses one or more such Experimental Radios.

Third-Party Blackbox Equipment (3PBBE): Incorporating these is one of the more ambitious aspects of the AERPAW architecture. In this category, we place any device (or a collection of devices that together form a vertical set, or ecosystem, of devices) that is essentially commercial equipment, meant for production use. A commercial hardware 4G or 5G base station is an example of this class of equipment. However, while such equipment is not intended to be used for experimental purposes, and thus typically designed without deep programmable or configurable interfaces and therefore difficult to experiment with, many experimenters would like to experiment with them nevertheless, for the realism of research results afforded by such devices.

At this time, the 3PBBE equipment integrated into the AERPAW platform consist of an Ericsson 4G/5G base station-RAN-core system, multiple Keysight RF sensors, and a system of four Facebook Terragraph radios.

Vehicle: Carrier of an APRN during an experiment during which the portable node acts as a mobile node. A vehicle may be programmable, providing experiment-controlled mobility (UAVs, Unmanned Ground Vehicles or rovers), or not, providing vehicle-controlled mobility (shuttles, SUVs, manually piloted UAS, backpacks on cyclists or pedestrians).

Virtual Computing Entities: In order to provide the experimenter with root access, but still retain supervisory override authorities (to preclude unsafe operations), we run multiple virtual computing entities in each AERPAW Node during Testbed execution. They can be instantiated as actual Virtual Machines (VMs) or containers; at this time AERPAW uses containers exclusively. In the rest of this paper, we use the terms VM and container interchangeably to indicate such virtual computing entities. AERPAW has two main virtual entities: **E-VM** and **C-VM**. The E-VM is the virtual computer afforded the experimenter, in which the experimenter enjoys root access. The experimental SDRs, and the UAV/UGV (more correctly its autopilot) are directly accessible from the E-VM by the APIs provided by standard programming libraries. Each

AERPAW Node also runs a C-VM container, whose function is monitoring and override; the functioning of the C-VM is articulated further below.

V. AERPAW Platform Control Framework

Having articulated the background that motivates the various execution modes, and their requirements, we are now in a position to appreciate that the architecture, design, and implementation of AERPAW's resource control mechanisms pose a rather unique scenario. In what follows, as we describe the control framework design characteristics, one aspect of the underlying philosophy (which is at variance with the control frameworks for many other platforms and facilities) is worth keeping in mind.

This is related to the idea of a *natural granularity of resources for control*. In any resource control problem, there are natural units of resource control (for example, in network orchestration, at one level of abstraction, individual routers or datapaths may be such natural resource units). Such units typically aggregate to form larger units of control at higher levels of aggregation. However, logically similar units (even if different in size and/or specifics of capability) are often amenable to the same level of control.

In the AERPAW architecture, it would be natural to assume that the **AERPAW Node** represents such a common unit of control, and resource control mechanisms should be first generically designed on the basis of exercising control on AERPAW Nodes, then specialized by considering differences in the various nodes. However, this turns out not to be true. The various kinds of AERPAW Nodes differ in more than details. Virtual nodes have different orchestration needs from that of physical nodes, and a very different set of underlying software environment. Orchestration of virtual nodes is much more amenable to implementation at single VE servers, while physical nodes are by definition each separate; the former are created and destroyed on demand, while the latter are permanently installed at specific locations, and each a unique resource. Indeed, from the resource control point of view, the VE itself is the resource to be controlled, not the virtual computing nodes they produce (as a result of controlled operation). AERPAW Portable Nodes are, in turn, different in their control needs and capabilities than AERPAW Fixed Nodes; portable nodes are exchangeable, but fixed nodes are not. Nevertheless, unlike virtual nodes, portable nodes are physical resources, so for a Testbed mode execution, specific portable nodes must be actually deployed in place of each virtual portable node the Experimenter developed code for - the distinction is that fixed nodes are bound a priori (since each is unique, and this uniqueness must be reflected in the emulation underlying the corresponding Development mode virtual nodes), while portable nodes are bound just in time for Testbed execution.

Accordingly, we have considered the issue of identifying the natural units for resource control in the specific context of AERPAW. The rest of this section presents the results of those considerations, among other things.

A. Control and Orchestration

"Control", in the platform context, refers to both control signaling, and a subset of management functions (FCAPS), in the traditional sense. Specifically, the most relevant capabilities that are necessary for Platform Control to address are:

1) Scheduling the usage of various resources, to support or for access by or on behalf of specific users of the platform,
2) Enabling and disabling access dynamically to various resources by use of appropriate credentials,
3) Dynamically configuring each resource to enable appropriate usage at any time.

While individual resources can be configured independently, platform target configurations require coordinated configuration of various resources. Only some combinations of configurations for a set of resources result in valid platform configurations; other combinations of configurations of the same set of resources (even if each is a valid configuration for that particular resource in some context) may result in a platform configuration that is not useful, and possibly vulnerable or otherwise undesirable.

The term "orchestration" is often used in this context to refer to the coordinated configuration of some target subset of resources to correctly achieve platform goals, while avoiding undesirable combinations of configurations even transiently. Thus the purpose of platform orchestration can be stated as the on-demand, coordinated, dynamic usage scheduling, access control, and configuration of various platform resources.

Automation of Orchestration: The term "automation" in this context refers to both (i) "softwarization" of specific control signaling and management decisions, (ii) use of additional software to implement a state machine in which certain conditions trigger other signaling/management actions without human mediation. Naturally, of all the resources (software, computing hardware, and non-cyber resources), only some may require (or even admit of) dynamic automated configuration or orchestration actions. For example, any manual safety actions taken by safety pilots to avoid unsafe operations by AERPAW UAVs during Testbed operation are inherently physical in nature, and are impossible to realize by an automated orchestration system. In contrast, other resources are infeasible or counter-productive to integrate into an automated orchestration system, such as weather reports or forecasts, even though they are informational elements, and could conceivably be integrated into an automated framework. The design decisions regarding which operations/resources to integrate into an automated framework, and which ones to enable in part by human-in-the-loop operations have to be taken on a case-by-case basis, weighing the penalties (complexity of software, development/maintenance overhead, cyber-resource requirements) against the benefits (reduced human effort, reduction in error-prone workflows, operational speed) in each case.

Therefore, deciding not to automate is as much a key design decision as to automate, for AERPAW design. To

understand these decisions in reference to specific elements of the platform's control functions, it is useful to think of two aspects of the control of each entity required to be controlled, in analogy with a person controlling any system. If the person corresponds to the control functions, then we note that this person's mind, or brain, takes the decisions regarding what control functions to exercise at what time. The person then uses hands and feet to implement that actual control. Sensory organs such as eyes and ears provide input both for the original control decision, and verification of the subsequent control actions.

For the platform, the "motor" functions are software, often glue code, that enables canonical functions of a platform element (such as the Virtual Environment, or individual AHNs, or containers) to be made available for easy operation. They often take the form of comparatively short scripts, that interact with a varied set of capabilities provided by more substantial pieces of software, or hardware drivers. (For example, a single short script of less than a hundred lines of code to enable setting up the Gateway side of a virtual mode experiment may interact with software or hardware firewalls, software or hardware VPN servers, container software, virtual switch software, and DHCP software.) The "sensory" functions are monitoring software, whether RF, vehicle, CPU, or memory, and also the support software to enable collection, storage, and query of gathered information. The "brain" functions are the software that can relate what input ("sensory" input or human input through Portal or other means) should trigger what "motor" actions.

In achieving operability of the platform, the "motor" functions are indispensable to implement. However, the "brain" implementation and integration can be deferred, if so necessitated by budget and schedule, because human Operators with actual brains can intentionally use any "motor" function that an automated "brain" would use. Of course, such human operation, while far more versatile, intelligent, and flexible, is far slower, and more prone to mistakes. Thus, to improve automation and scalability, the "human brain" based operation needs to be a temporary and short phase, simply to allow validation of projected control flow, and also to allow budget and schedule to catch up, so that the "brain" functions can be increasingly implemented. Here and elsewhere, it is the collection of all such "brain" functions that we refer to collectively as the "**AERPAW Platform Control Framework**" or "Control Framework" (APCF, PCF, or simply CF).

Thus, in the AERPAW context, APCF actions relate *exclusively* to automated orchestration. All orchestration (in the sense defined above) is part of AERPAW Operations, but *only automated aspects are part of the APCF*.

B. Resources

To understand the design of the software that enables control over various parts of the AERPAW platform, it is helpful to start with a broad view of the platform resources, including non-software resources, and resources that will not be controlled or managed by software.

Variety of Resources: The entire AERPAW facility consists of resources (software, hardware, and non-cyber) that can be viewed as follows. Some resources are highly specific to, and dedicated to realizing, one of the four canonical experiment execution environments (Development, Emulation, Sandbox, Testbed) :

- AFRNs, APRNs, (Testbed, Sandbox)
- Third-Party Black Box Equipment (Testbed, Sandbox)

Some resources enable or realize platform capabilities that underlie and cut across multiple of the above environments, and can themselves be considered complex resources, specifically:

- Virtual Experiment Environment,
- Secure Backplane,
- Experimenter's Web Portal,
- Operator's Experiment Oversight,
- Monitoring Service,
- AERPAW Operations Environment.

Hardware computing and communications resources, i.e. servers/computers, switches, fiber plants, radio equipment, antennae, provide the substrate for the above. Software resources such as Operating Systems, or other general-purpose software packages, are also resources that provide the substrate for custom AERPAW software resources.

Beyond computing and communications hardware and software, general physical resources such as vehicles, space, racks, power, heating/cooling/conditioning, towers, enclosures, cables, are also required. Finally, human resources such as safety pilots or operators are also involved in enabling the dedicated or shared resources identified above.

Hierarchy of Resources: From the above, it is clear that resources can be considered at various levels of abstraction.

At the lowest levels, we might consider individual computers, switches, OSs, rotors, radios, as **Basic Resources**. Despite this terminology, these resources are typically quite complex constructs themselves. The main distinction of a Basic Resource is that they are general-purpose (non-AERPAW-specific), and procured from providers or vendors as generic off-the-shelf products or resources available "as-is". The term "basic" in this context refers, therefore, to the fact that they are elemental building blocks from the point of view of AERPAW fabricators.

In contrast, any resource that is AERPAW-specifc, and custom-designed or custom-assembled from Simple Resource (and/or other Complex Resource) building blocks, are termed **Complex Resources**.

The term "fabrication" is used to indicate such configuring, combining, or assembling Basic Resources (generic) into Complex Resources (AERPAW-specific). Such fabrication is expected to be one-time (or at least infrequent) activities. Examples include the design and implementation of databases, ARNs, AERPAW vehicles, SDR front-ends.

The term "operation" is appropriate for smaller, more atomic actions executed moment-by-moment on a resource, to leverage the actual capabilities of that resource. Examples include sending specific commands to a vehicle, querying a database or modifying an entry in it.

The term "configuration" is more appropriate when a basic or complex resource is being put into some particular configuration or state in preparation for an extended, but short, time of operation in a given context. Examples include configuring an ARN for the testbed mode execution of a particular experiment, or configuring the Secure Backplane to create custom connectivity amongst a specified subset of ARNs and other resources for the same.

That is, configuration implies readying a resource for use, and operation implies using it.

Abstraction of Resources: Complex Resources abstract the Basic Resources that constitute them, obviously. Once a Basic Resource (say, a rotor, or a network switch) has been assembled into a Complex Resource (say, an AERPAW UAV, or the Secure Backplane), (i) use (operation) of the Basic Resource is only permitted through use of the Complex Resource, and (ii) only the Complex Resource is privileged to configure the Basic Resource. For all external entities, the Complex Resource is the only one that can be configured and used. Such abstraction stems from **encapsulation**.

A second kind of abstraction is possible, when a Complex Resource is tasked (as part of its function) with the complete responsibility of configuration and operation of some other resource, but the other resource remains a distinct resource, and is not composed or assembled into the Complex Resource itself. In such a case, the abstraction is an architectural choice; potentially to distribute functionality or complexity in the system, or from considerations of ease of development, deployment, or maintenance. This type of abstraction is driven by **delegation**, and future ongoing design and development may choose to create additional alternate pathways of configuration or operation to such abstracted resources.

C. PCF Mission Identification and Scope

The APCF, as defined before, is the "brain" for all automated orchestration actions; as we defined immediately above, not all resources are visible to the PCF (some are encapsulated or delegated); finally, we remarked above that not all resources are amenable to automated orchestration actions (some orchestration actions for some resources are planned to be manual, by physical constraint, or by design). Thus the scope and responsibility of the APCF is limited to some orchestration actions for some AERPAW resources.

Below we list the Complex Resources that are of relevance to the APCF, in that the APCF will be required to automate some or all orchestration actions for them:

- Virtual Experiment Environment
- AERPAW Radio Nodes (AFRNs, APRNs)
- Secure Backplane
- Portal front-end

These Complex Resources abstract the relevant configuration capabilities of various other resources, so that the APCF does not have to control the configuration of those resources separately. The Virtual Experiment Environment abstracts, among other things, docker containers, Linux (OVS) bridges, network interfaces, VLAN stitching, IP address assignment, purpose-built AERPAW software for emulation, vehicle control, SDR control. Each AFRN/APRN companion computer abstracts: containers, SDRs, vehicles, VLAN stitching, IP address assignment, purpose-built AERPAW software for emulation, vehicle control, SDR control. The Secure Backplane abstracts VLAN stitching, DHCP, OVPN, Third-party Black-Box Equipment reachability and isolation. Portal front-end abstracts Experimenter and Operator interactions, Experimenter credential management, Experiment and Operator authentication.

APCF Requirement: The main purpose of automation (of orchestration) is to reduce the delay and error-proneness that is invariably associated with manual operation (and thereby reduce drudgery for human operators), and improve identical repeatability, of repeated tasks. It follows that the orchestration actions most necessary to automate are those that are most likely to be repeated, in a predetermined and predictable manner. Attempting to automate workflows that are not expected to be performed frequently, or require customization each time they are performed, does not improve efficiency or effectiveness of operations. We do not undertake a detailed description of the workflows here, but they have been explicitly identified and described in the AERPAW User Manual [16], and also addressed in other publications such as [20].

The Experimenter's side of such a workflow relates largely to interactions with a web-based Portal. The Experimenter also interacts with the Virtual Experiment Environment, by logging in to perform development tasks, but these are inherently "use" interactions, not configuration actions (and are driven by the human Experimenter's interactive sessions), and therefore not a target of platform automation.

The bulk of the coordinated configuration tasks that are required to be performed to support this workflow is on the AERPAW Ops's side of the interactions. A detailed description of the actual actions that have to be performed by AERPAW Ops is in the AERPAW Summary Ops Manual, and is not repeated here for the sake of brevity.

The mandate of the APCF, therefore, can be stated as follows:

> "Map the various Experimenter and Operator actions in the canonical AERPAW experiment workflow to the corresponding sets of configuration capabilities of the relevant AERPAW Complex Resources, and trigger these configuration actions automatically."

It remains only to identify the specific configuration capabilities of the Complex Resources that the APCF needs to orchestrate. For the Portal front-end, they are the interactions to/from the Experimenters and the Operators that have to be used to trigger APCF actions.

- Virtual Experiment Environment
  - Instantiate/Destroy virtual experiment for Development mode - containers, bridges, addresses
  - Save/Load E-VM contents for Development mode virtual experiment

  - Instantiate/Destroy virtual experiment for Testbed mode - containers, bridges, addresses
- Secure Backplane
  - Create/Destroy custom DHCP service, OVPN gateway, XM plane for virtual experiment
  - Stitch/Unstitch XM plane at virtual experiment execution server for virtual experiment
  - Create/Destroy custom DHCP service, OVPN gateway, XM plane for testbed experiment
  - Stitch/Destroy XM plane at virtual experiment execution server for testbed experiment
  - Create/Destroy network planes, stitch XM plane at AFRN for testbed experiment
  - Create/Destroy planes, stitch XM plane at APRN for testbed experiment
  - Move E-VM contents to/from virtual environment server to AFRN/APRN
- AERPAW Radio Nodes
  - Instantiate testbed experiment - containers, software
  - Load/Save E-VM contents into/from containers
  - Start/Stop Testbed experiment execution
- Portal front-end
  - Experimenter create Experiment
  - Experimenter instantiate Development session
  - Experimenter save / save-and-exit Development session
  - Experimenter submit Experiment for Testbed session
  - Operator update status of Experiment

This concludes the AERPAW-specific aspects of the control framework considerations. While many other functionalities have to be implemented by the control framework, such as user management, access control, calendar management, etc., they are largely similar for AERPAW as for many other testbed facilities, and we do not discuss their design or technology implementation in this paper.

ACKNOWLEDGMENT

This material presented in this paper is based upon work supported by the National Science Foundation PAWR Program under Grant No.: CNS-1939334, and its supplement for studying NRDZs. We also gratefully acknowledge colleagues on the AERPAW team, and other colleagues in the community, who helped articulate and crystallize research needs, and meaningful operational models, for AERPAW's original and ongoing testbed vision. We gratefully acknowldge the industry partners who provided equipment as in-kind support to be used in AERPAW as 3PBBE, and allowed their names to be used on the AERPAW website, in this paper, and elsewhere.

VI. CONCLUSION

We have presented an overview of the AERPAW Platform Control Framework, with context that provides insight into the motivation, ambitions, and constraints for its design. In future, we plan to publish the evolution of this architecture, as well as comparisons with equivalent mechanisms in other physical testbeds. We hope this exposition may inform designers and architects in a broad range of non-conventional cyber-physical research facilities.

REFERENCES

[1] SESAR, European Union, Euro Control, "European drones outlook study: Unlocking the value for Europe," Nov. 2016. [Online]. Available: {https://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf}

[2] AUVSI, "The Economic Impact of Unmanned Aircraft Systems Integration in the United States," Mar. 2013. [Online]. Available: https://higherlogicdownload.s3.amazonaws.com/AUVSI/958c920a-7f9b-4ad2-9807-f9a4e95d1ef1/UploadedImages/New_Economic%20Report%202013%20Full.pdf

[3] Goldman Sachs, "Drones: Reporting for Work." [Online]. Available: https://www.goldmansachs.com/insights/technology-driving-innovation/drones/

[4] J. Desjardins, "Amazon and UPS are betting big on drone delivery," Business Insider, Mar. 2018. [Online]. Available: https://www.businessinsider.com/amazon-and-ups-are-betting-big-on-drone-delivery-2018-3

[5] "Amazon Prime Air Drone Delivery." [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011

[6] "Google Project Wing." [Online]. Available: https://x.company/projects/wing/

[7] "Uber Elevate." [Online]. Available: https://www.uber.com/us/en/elevate/

[8] J. Johnsson and A. Levin, "Boeing is getting ready to sell flying taxis," Bloomberg, Mar. 2018. [Online]. Available: https://www.bloomberg.com/news/articles/2018-03-01/boeing-is-getting-ready-to-sell-flying-taxis-within-a-decade

[9] "Zipline medical supply drone delivery." [Online]. Available: http://www.flyzipline.com/

[10] "Flytrex on demand drone delivery." [Online]. Available: http://www.flytrex.com/

[11] "Matternet delivery drones." [Online]. Available: https://mttr.net/

[12] Art Pregler, AT&T, "When COWs Fly: AT&T Sending LTE Signals from Drones," Feb. 2017. [Online]. Available: https://about.att.com/innovationblog/cows_fly

[13] Vodafone, "Beyond visual line of sight drone trial report," Nov. 2018. [Online]. Available: https://www.vodafone.com/content/dam/vodafone-images/media/Downloads/Vodafone_BVLOS_drone_trial_report.pdf

[14] X. Lin, V. Yajnanarayana, S. D. Muruganathan, S. Gao, H. Asplund, H.-L. Maattanen, M. Bergstrom, S. Euler, and Y.-P. E. Wang, "The sky is not the limit: LTE for unmanned aerial vehicles," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 204–210, 2018.

[15] Qualcomm, "LTE Unmanned Aircraft Systems," May 2017. [Online]. Available: https://www.qualcomm.com/media/documents/files/lte-unmanned-aircraft-systems-trial-report.pdf

[16] "AERPAW User Manual." [Online]. Available: https://sites.google.com/ncsu.edu/aerpaw-wiki/

[17] "Aerial Experimentation and Research Platform for Advanced Wireless." [Online]. Available: https://aerpaw.org

[18] A. Panicker, O. Ozdemir, M. L. Sichitiu, I. Guvenc, R. Dutta, V. Marojevic, and B. Floyd, "Aerpaw emulation overview and preliminary performance evaluation," *Computer Networks*, vol. 194, p. 108083, 2021.

[19] V. Marojevic, I. Guvenc, R. Dutta, M. L. Sichitiu, and B. A. Floyd, "Advanced wireless for unmanned aerial systems: 5g standardization, research challenges, and aerpaw architecture," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 22–30, 2020.

[20] M. Mushi, H. Joshi, R. Dutta, I. Guvenc, M. L. Sichitiu, B. Floyd, and T. Zajkowski, "The aerpaw experiment workflow - considerations for designing usage models for a computing-supported physical research platform," in *Proceedings of the 9th International Workshop on Computer and Networking Experimental Research using Testbeds (CNERT), organized in conjunction with INFOCOM 2022*, May 2022.

[21] T. Samal, R. Dutta, I. Guvenc, M. L. Sichitiu, B. Floyd, and T. Zajkowski, "Automating operator oversight in an autonomous, regulated, safety-critical research facility," in *Proceedings of the 31st International Conference on Computer Communications and Networks (ICCCN 2022)*, July 2022.