Network Intelligence for Virtualized RAN Orchestration: The DAEMON Approach

Marco Gramaglia, Miguel Camelo, Lidia Fuentes, Joaquín Ballesteros
Gabriele Baldoni, Luca Cominardi, Andres Garcia-Saavedra, Marco Fiore

***Abstract*—Next-generation mobile networks will largely benefit from advances in softwarization and cloudification of network functions. However, fully exploiting the new potential of flexible network architectures in front of increasingly demanding service volumes and requirements calls for an extremely effective integration of Network Intelligence (NI) solutions into production infrastructures. While current standardization efforts towards embedding NI in beyond-5G and 6G systems are still in their infancy, the DAEMON project is developing technologies for a NI-native generation of mobile networks. In this paper, we present current evolutions proposed by DAEMON in terms of a general model for the representation of NI instances, which facilitates their synergic integration in network environments. We showcase the practical viability and advantages of the proposed approach with two state-of-the-art NI algorithms for vRAN orchestration implemented into an open-source data flow programming framework.**

***Keywords*—vRAN, Virtualized RAN, Network Intelligence, Mobile Networks, Orchestration, 6G**

I. Introduction

With the current worldwide deployment of the fifth-generation (5G) and forthcoming sixth-generation (6G) mobile networks, the wireless communications community has started looking into novel architectures to support the current softwarization and cloudification trends [1]. Innovative network architectures will require advanced Artificial Intelligence (AI) and Machine Learning (ML) algorithms, executed by heterogeneous orchestrators and controllers to manage various micro-domains or network slices. These algorithms constitute the intelligence of the network, i.e., Network Intelligence (NI), being capable of dynamically taking actions according to a service request or fluctuations in network activities.

In this context, it is expected that different NI instances will be deployed across the network to solve a variety of networking tasks such as end-to-end orchestration and management, system control, network service monitoring and analysis, among others. Each of these instances will adhere to numerous Key Performance Indicator (KPI) targets, including Quality of Service (QoS) or Quality of Experience (QoE) guarantees, maximization of infrastructure and resource reuse across different tenants or network services, and full network automation to achieve zero-touch network and service management.

An example of a network domain that has evolved to embrace such a vision is the Radio Access Network (RAN), where recent advances in Network Function Virtualization (NFV) and Software-Defined Networking (SDN) have spearheaded flexible and scalable Radio Access Network virtualization (vRAN) deployments. As a consequence, the concept of RAN Intelligent Controller (RIC) has arisen, a novel architectural component that provides a centralized abstraction of the network, where network operators can design, implement, and deploy custom control-plane Virtual Network Functions (VNFs) to perform RAN optimization via closed control loops at different timescales aided by ML. This is fundamental since present implementations of algorithms that perform RAN optimizations are far from optimal [2]. For example, they inefficiently pool computing resources and just over-dimension computational capacity to cope with peak demands in real-time workloads [3] or do not consider energy constraints to deploy small cells [4].

However, the current architectural and VNFs design does not provide specifications or guidelines to include NI yet. Moreover, the present efforts promoted by major standardization bodies towards the integration of NI in next-generation network architectures are still in their infancy [5], [6]. The mobile network architecture proposed in the context of the DAEMON project[1] is NI-native and goes several steps beyond the current standardization trends [5], [7]–[9]: it posits a new approach for systematic integration of NI in 6G infrastructure while staying entirely aligned with emerging designs in standardization [2]. In our previous work [10], we outlined general requirements and specifications for NI design that stem from data management, diverse control timescales, and network technology characteristics; we also derived initial principles for the design of an NI Orchestration layer, which focuses on (i) proposals for the interaction loop between NI instances and the NI Orchestrator, and (ii) a unified representation of NI algorithms.

In this paper, we build on the concepts introduced in [10] and exploit DAEMON unified representation of NI algorithms to integrate NI in the context of vRAN orchestration and control problems. The latter is an appropriate and representative context for the embedding of NI solutions into a larger system because (i) they deal with high-dimensional state spaces (including, e.g., channel quality patterns or network load patterns across a potentially large number of users and base stations), and (ii) orchestration and control decisions are often coupled among a large number of parameters (for instance, radio and computing resource policies). In these settings, finding a suitable framework that maximizes the reuse of ML models across diverse NI solutions is of paramount importance: it paves the way for a structured approach to ML assimilation in networking systems, greatly simplifying future innovations in NI for beyond-5G and 6G networks.

M. Gramaglia is with University Carlos III of Madrid. M. Camelo is with University of Antwerp - imec, IDLab. L. Fuentes and J. Ballesteros are with ITIS Software, Universidad de Málaga. G. Baldoni adn L. Cominardi are with ADLINK Technology. A. Garcia-Saavedra is with NEC Laboratories Europe. M. Fiore is with IMDEA Networks Institute.

[1] https://h2020daemon.eu/

To cite this article, please refer to the published version:
M. Gramaglia et al., "Network Intelligence for Virtualized RAN Orchestration: The DAEMON Approach," 2022 Joint European Conference on Networks and Communications & 6G Summit(EuCNC/6G Summit), 2022, pp. 482-487.
DOI: 10.1109/EuCNC/6GSummit54941.2022.9815816

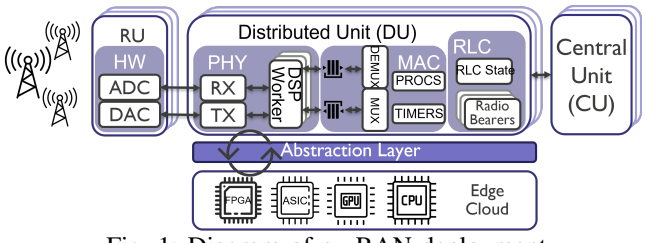

Fig. 1: Diagram of a vRAN deployment

Specifically, we present mappings between the N-MAPE-K loop proposed as the unified NI representation within DAEMON [10] and two sample NI algorithms developed in the project: `vrAIn` [11] and `SBP-vRAN` [12]. We show that the DAEMON architecture promotes the reuse of NI components, such as monitoring ones, among heterogeneous algorithms used inside each micro-domain. We then demonstrate the DAEMON approach by implementing the unified framework in Zenoh-flow[2], a dataflow-based programming platform that is especially suited to support the operation of AI algorithms with different time scales such as end-to-end deadlines, automatic timestamps, or feedback support.

The document is structured as follows. Section II presents NI in the context of vRAN. In Section III, we describe the two state-of-the-art ML-based algorithms for vRAN optimization that are considered in our study. We define their specific requirements as NI algorithms, and we map them to the unified representation of NI proposed in [10] in Section IV. Finally, Section V realizes them via a framework for data flow programming to demonstrate its generality, and Section VI discusses conclusions and future work.

II. Network Intelligence for vRAN

vRAN is well-recognized as a key technology to accommodate the ever-increasing demand for mobile services at an affordable cost for mobile operators. vRAN centralizes softwarized base stations into common computing infrastructure in a cloud location (typically at the edge) via NFV. Figure 1 illustrates a set of base stations sharing a common pool of computing resources to perform radio processing tasks such as signal modulation and forward correction coding (FEC). This provides several advantages, such as resource pooling (via centralization), simpler update roll-ups (via softwarization), and cheaper management and control (via commoditization), leading to savings of 10-15% in capital expenditure per km2 and 22% in CPU usage [13]. It is thus not surprising that vRAN has attracted the attention of academia and industry. OpenRAN , O-RAN, or Rakuten's vRAN—led by key operators (such as AT&T, Verizon, or China Mobile), manufacturers (such as Intel, Cisco, or NEC), and research leaders (such as Standford University)—are examples of publicly disseminated initiatives towards fully programmable, virtualized and open RAN solutions based on general-purpose processing platforms.

In addition to the adoption of NFV, one of the most recent innovations introduced in the context of open RAN is the programmable components to optimize the performance of the system in a closed-loop approach. A realization of this innovation are the near-Real-Time (RT) RIC and non-RT RIC

[2]https://github.com/eclipse-zenoh/zenoh-flow

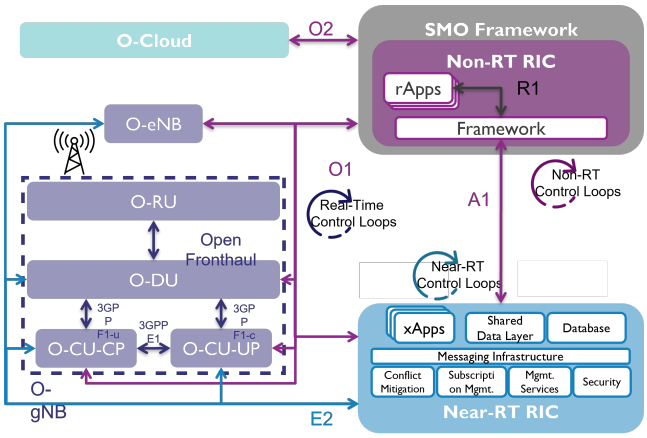

Fig. 2: The O-RAN architecture

proposed by the O-RAN alliance, as shown in Figure 2. These two logical controllers are used to create a centralized view of the network by consuming monitoring data generated from the network infrastructure (e.g., number of users, resource utilization, etc.). The algorithms running on these controllers are expected to be empowered by AI and ML techniques (i.e., NI), which introduces a data-driven approach to optimize the network performance in a closed-loop fashion automatically.

The non-RT-RIC is integrated into the network Service Management and Orchestration (SMO) layer and operates on a timescale larger than 1 second. Its main goal is to support intelligent RAN optimization by providing policy-based guidance, ML model management, and enrichment information to the near-RT RIC function so that the RAN can optimize, e.g., Radio Resource Management (RRM) under certain conditions. It can also perform intelligent RRM function in non-real-time interval (i.e., greater than 1 second). In a shorter operation time scale, the near-RT RIC is a logical function that enables near real-time control and optimization of the RAN and its resources via fine-grained data collection and actions over open interfaces and with control loops in the order of 10ms-1s. It hosts one or more xApps, an application designed to run on the near-RT-RIC, to collect near real-time information (e.g., on a User Equipment (UE) or a Cell basis) and provide control over the RAN. The control is steered via the policies and the enrichment data provided from the Non-RT RIC.

While the near-RT RIC hosts the xApps as mentioned before, the non-RT-RIC hosts the rApps, which are the functions that are used to provide value-added services to support and perform RAN optimization and operations. More generally, both functions are basically custom logic to perform RAN optimization, told apart by the fact that they operate at different time scales. When such r/xApps are empowered by AI/ML algorithms, they match the definition of NI [10]. However, to integrate NI natively in next-generation network architectures, such as the AI/ML-based r/xApps in O-RAN, a large range of NI instances will need to interact seamlessly to perform at their best, and exchange data and information to mutually improve both their learning and decision-making processes. For this, it is fundamental that NI for vRAN optimization can support closed-loop control, and it can be defined via a well-defined modeling order to create components that can be easily extended, re-usable, and easily migrated among platforms that support NI by design.

III. vRAN NI Algorithms

We next present two algorithms developed within the context of the DAEMON project and fully detailed in [11] and [14], respectively, which handle vRAN orchestration and control operations at different timescales. We will use these algorithms as an example of the approaches that may be deployed concurrently, which may benefit from the framework proposed in this paper.

A. `vrAIn`: Radio & computing resources joint orchestration

`vrAIn` [11] is a vRAN orchestration framework that is composed of two main modules: (i) the CPU and Radio schedulers, which provide the radio access functionality operating at low timescales (milliseconds), and (ii) the resource manager, which controls the behavior of such schedulers at larger timescales (seconds). Through them, `vrAIn` optimizes VRAN operation in terms of throughput and resource (CPU) utilization. The overall `vrAIn` framework is depicted in Figure 3.

Schedulers. Two main tasks are performed by these schedulers: (i) selecting the Modulation and Coding Scheme (MCS) for a given set of users depending on *e.g.*, their channel quality or they expected load and (ii) assigning the decoding task (the most expensive operation of a vRAN system) of a subframe to the available CPUs. Both of them operate at submillisecond timescales, according to the following criteria: (i) a *maximum computing time fraction* $c_i \in \mathcal{C} := [0, 1] \subset \mathbb{R}$ (also called **computing control decisions**); and (ii) a *maximum MCS* $m_i \in \mathcal{M}$, where $\mathcal{M}$ is a discrete set of MCSs (also called **radio control decisions**).

Resource Manager. This module implements the artificial intelligence techniques described in [11] to perform resource assignment decisions on larger timescales (*i.e.*,, seconds). Specifically, it implements a feedback control loop that: (i) analyzes *contextual* information (Signal-To-Noise-Ratio (SNR) and traffic patterns); (ii) Enforces the learned CPU and radio control *policies*, which map contexts into schedulers control decisions by using the interfaces introduced above; and (iii) Assesses the quality of the taken decisions by analyzing a *reward* signal, designed to maximize performance.

To perform the tasks discussed above, we employed in [11] a Reinforcement Learning algorithm based on contextual bandits, which solves outstanding challenges such as the space state compression through autoencoders deployed within an xApp in the near-RT RIC, and a deep deterministic policy gradient (DDPG) algorithm implemented with an actor-critic neural-network structure that is implemented as a rApp in the non-RT RIC.

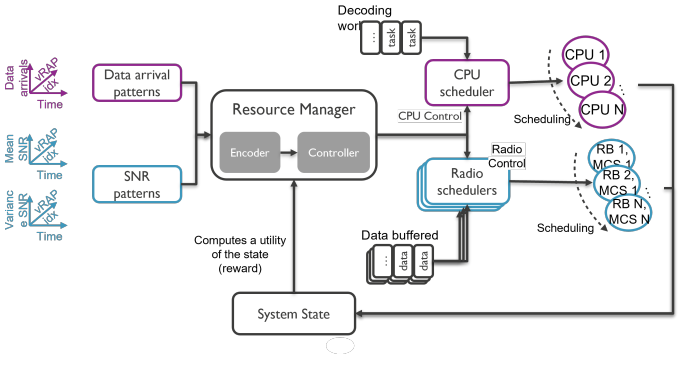

Fig. 3: `vrAIn` system design

B. `SBP-vRAN`: Energy-driven RAN control

We now consider a near-real-time xApp to control vBSs deployed in an energy-constrained cloud infrastructure. This type of BSs is relevant for low-cost small cells, Power-over-Ethernet (PoE) cells, and similar platforms increasingly common in 5G-and-Beyond networks. Our goal is to use O-RAN's control architecture to implement near-real-time configuration policies that are adaptive to system dynamics while satisfying *hard* energy constraints. Specifically, we consider the Safe Bayesian Optimization vRAN control algorithm (`SBP-vRAN`) recently introduced in [12], [14].

Context Information. `SBP-vRAN` defines the downlink (DL) context at each period t, which includes the mean and variance of the DL channel quality indicator (CQI) across all users in the previous period, and the *new* bit arrivals at the vBS DL aggregated across all users. The uplink (UL) context comprises the mean UL SNR across all users, and the new UL bit arrivals are estimated from the periodic Buffer Status Reports of the users. All these measurements are collected by the Near-RT RIC's Data Monitor from the vBS using the E2 interface at the sub-second granularity and are aggregated at the start of each control period t.

Actions. Every period t, `SBP-vRAN` makes a series of decisions corresponding to radio-related configuration policies. For the downlink, these include *transmission power control (TPC) policy* to control the maximum allowed vBS transmission power, the highest MCS eligible by the vBS (*DL MCS policy*), and the maximum vBS transmission airtime (*DL airtime policy*). Conversely, the uplink policies include an UL MCS and airtime policies. Once computed, the xApp deploys the policy into the vBS through the E2 interface.

Rewards. `SBP-vRAN` rewards a fair distribution of throughput performance across all users in both uplink and downlink. It is important to stress that in practice, we can only have **noisy values** of this reward function, even when its arguments are fixed, because, at such fast timescales, the system is naturally stochastic. Power consumption measurements are also noisy to make the problem even more involved.

`SBP-vRAN` is specifically designed to handle such impairments. The goal of `SBP-vRAN` is to find maximal-throughput configurations that also respect the available power budget. `SBP-vRAN` achieves this goal by employing a *safe* exploration of the configuration space to satisfy the power threshold at any period, i.e., not only at the final optimal-operation stage. To solve this problem, `SBP-vRAN` resorts to a non-parametric learning approach using Gaussian Processes, Contextual Bandits, and Bayesian learning. The approach has the additional practical advantage that one can change the power budget in run time without restarting the learning process. Other parametric methods, such as Reinforcement Learning relying on neural networks, need to be re-trained if the constraint changes, which substantially increases the required training data. The details of the learning model can be found in [12], [14].

IV. A UNIFIED FRAMEWORK FOR NI AUTOMATION

To ensure that diverse NI algorithms such as those presented above can coexist and run effectively in the same mobile network infrastructure, we need to abstract their complexity and produce a homogeneous representation of their multi-timescale operation. To address this challenge, we propose a split between the NI algorithms' requirements and the interactions between their different components. We will discuss this approach in detail next.

A. Requirements

The vRAN orchestration algorithms discussed in Section III introduce two main functional requirements: monitoring capabilities (identified as *FR-001* in Table I) and resources allocation capabilities (identified as *FR-003.x*). *FR-001* specifies the need for NI solutions that provide monitoring capabilities to obtain information about the current state or context of the system. Given this requirement, our analysis has identified relevant metrics such as CPU load, wireless channel conditions (SNR, CQI), traffic demands (as Buffer State Reports from the terminals and downlink buffer occupancy), or power consumption measurements. This also describes the need to integrate sensing capabilities in vRAN systems. RAN systems should provide APIs to access raw data from all the layers in the radio stack. We remark that Open RAN solutions specified by O-RAN are a key enabler to this end.

From *FR-001*, another functional (*FR-002*) and one non-functional (*NFR-001*) requirement are derived. On the one hand, *FR-002* describes the need to reduce the dimensionality of the state/context space and provide an expressive latent representation that is relevant to take appropriate actions. On the other hand, *NFR-001* considers advanced NI solutions that allow dynamic change of sources and time scales in the monitored data at runtime.

The last pillar of requirements is the allocation of resources and policies. We conclude that NI solutions should offer the ability to allocate resources as per *FR-003* dynamically. Specifically, computing resources (*FR-003.001*) and radio resources (*FR-003.002*).

Table I summarizes how each of those functionalities is associated with the `vrAIn` and `SBP-vRAN` algorithms described in Section III. Note, however, that any NI solution for the orchestration of vRAN that leverages the architecture discussed in Section IV will build on the same set of functionalities to work.

B. N-MAPE-K

Besides the specific requirements associated with the algorithms, as discussed in Section IV-A, we need a mechanism to create a common framework to map the most common features of NI algorithms, subsequently integrate them into the overall architecture, and design the necessary interfaces that algorithms use to interact with their environment.

For this purpose, we adopt within the DAEMON project activities a methodology already used by the MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) feedback loop–one of the most influential reference control models for autonomic and self-adaptive systems [15]. The methodology

TABLE I: Main functional and non-functional requirements association into specific vRAN NI algorithms. The nomenclature is that adopted to label NI requirements within the DAEMON project [1].

	`vrAIn`	`SBP-vRAN`
FR-001	API to measure CQI, BSR, and CPU load	API to measure SNR, CQI, BSR, downlink buffers, and power consumption
FR-002	Reduce dimensionality of state space	Reduce dimensionality of state space
NFR-001	Provide state information in a timescale of 1 second	Provide state information in a timescale of 100 ms
FR-003.001	Ability to allocate CPU resource shares to individual BS	-
FR-003.002	Ability to deploy MCS policies	Ability to deploy MCS, tx power, and airtime policies

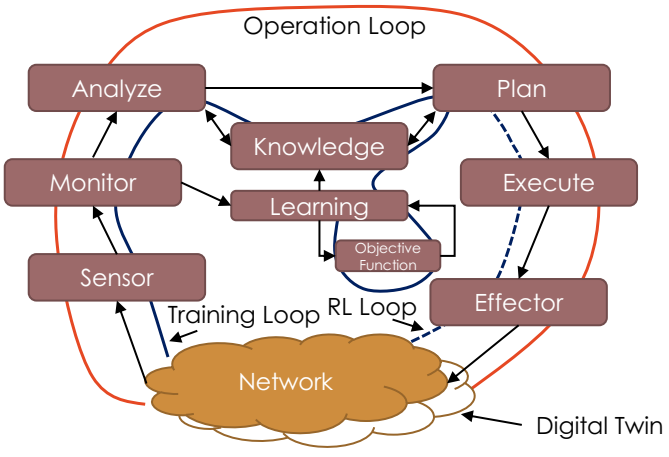

Fig. 4: Extended N-MAPE-K abstractions for NI algorithms

was first introduced in [10], and it allows classifying the algorithms that run at NI instances in a unified manner, based on how they interact with the other elements of the network.

It is worth noting that the original MAPE-K framework has limitations in the target context of mobile network functionalities supported by NI. Therefore, we propose changes to the legacy MAPE-K to take into account the specificities of the network environment, as depicted in Figure 4, to build the Network MAPE-K (N-MAPE-K). Specifically, we extend MAPE-K along two dimensions:

- The purpose of the NI, whether the Knowledge is being trained or used in inference for the operation of the network, following the MLOps paradigm.
- The nature of NI algorithm, distinguishing between supervised learning and reinforcement learning.

For the latter, the knowledge module shall be integrated with a **Training** definition, which specifies aspects such as the input data shape, batches, and most importantly, the used loss function (which could be dynamically adjusted) and the State/Action representation. Additionally, the effector and the sensors can also be redirected to a **Digital twin** element if needed by the specific NI instance.

With this framework, already presented in [10], we can represent the two algorithms discussed in Section III in a

unified way. This is summarized in Table II, which maps each phase of `vrAIn` and `SBP-vRAN` into the N-MAPE-K model.

V. Implementation in Zenoh-flow

The formal description of NI algorithms regarding their requirements and N-MAPE-K operation outlined in the previous Sections offers clear advantages in terms of integration into network infrastructures. Indeed, it produces a consistent and compact representation model that can be applied to inherently diverse solutions, such as `vrAIn` and `SBP-vRAN`. In turn, operators can benefit from such a representation by easily identifying the common needs of the different NI instances (expressed by matching requirements), or by merging operational phases that are common to multiple NI algorithms (e.g., monitoring functions in the case of the two sample algorithms in Table II.

In addition, our proposed representation can simplify the practical integration of NI instances into real-world systems. To prove our point, we next demonstrate how the sample `vrAIn` and `SBP-vRAN` algorithms can be easily implemented into an operational data flow programming framework once they have been mapped into an N-MAPE-K model.

A. The Zenoh Flow framework

Zenoh Flow provides a data flow programming [16] framework to ease the development of any application that requires cloud-to-thing data flows. In data flow programming, an application is divided into simple tasks, called *Operators*. Such operators are arranged in a graph, and the interconnections between them represent the application; then, each operator can execute concurrently with substantial parallelization gains [17]. By following such an approach, Zenoh Flow is designed to deliver the performance and efficiency required by control-oriented applications while supporting higher-level abstractions needed for some machine learning and AI data flows. An initial implementation of Zenoh Flow is published as open-source software (see footnote 2).

Cloud-to-Thing compatible. Zenoh Flow is designed to cope with the Cloud-to-Thing continuum. Specifically, Zenoh Flow abstracts the underlying fabric through Zenoh's unified API. Therefore, Zenoh Flow can seamlessly run a data flow graph across multiple machines, enabling migration, load balance, and redundancy for operators composing the graph.

Feature-rich. Zenoh Flow provides a set of features to facilitate the creation and management of operators. Example of such features are *automatic timestamps* and *end-to-end deadlines*. In Zenoh Flow, whenever data "enters" the data flow, it gets timestamped. Such timestamp can be used on each operator, and it is propagated throughout the graph. Developers have the possibility to define deadlines along any arbitrary path of the graph. When a deadline is missed, the last operator is notified at the end of the deadline. Zenoh Flow accepts *input rules*. Input rules are a way for the developer to specify under which condition the operator can be triggered. For example, it is possible to trigger the computation only if a subset of the inputs is present or if the inputs are synchronized. Finally, Zenoh Flow allows defining *loops* in the graph. Loops are helpful for applications that require *feedback*. An example of an application that involves feedback is a control loop. Network orchestration falls directly in that category with operators that rely on AI/ML to perform analysis or predictions.

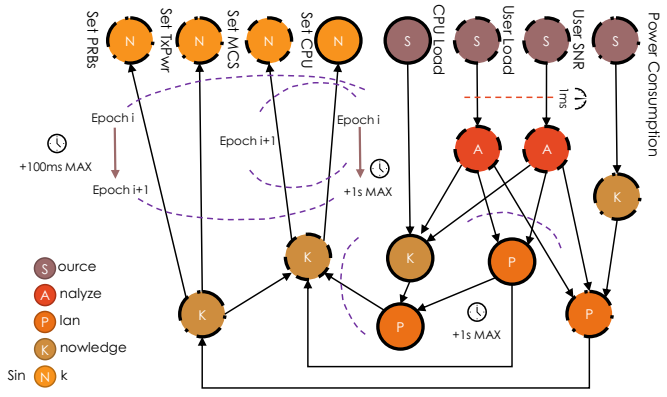

Fig. 5: The Zenoh-flow arrangement of the N-MAPE-K blocks for `vrAIn` and `SBP-vRAN`. Dashed nodes are common to both algorihtms, while solid and dot-dashed nodes are related to `vrAIn` and `SBP-vRAN`, respectively

Reusable. Zenoh Flow encourages code reuse, as operators can be developed independently and then used to compose different data flow graphs and, therefore, different applications. Providing reusable operators offers many advantages: users could publish them in a library, which could then be leveraged to develop applications faster and increase user adoption.

B. Implementation of NI algorithms

Figure 5 shows a possible mapping of our target vRAN orchestration algorithms into Zenoh Flow. The topmost part of the graph depicts the Sources and the Sinks, that implement the Monitor / Sensor and the Execute / Effector in Zenoh Flow. In a nutshell, these are the input and output variables of the NI algorithms, which in this case, are shared by the two NIs.

Four sources provide the input to the two algorithms: the user SNR and BSR reports (common to `vrAIn` and `SBP-vRAN`), the CPU load (only used by `vrAIn`), and the Power Consumption (only used by `SBP-vRAN`). Thanks to the Zenoh Flow features, these sinks can produce the data needed for the NI algorithms at a fixed pace (e.g., every TTI, that is, at intervals of 1 ms) or on-demand. The former is the case of SNR (in both uplink and downlink direction) and user load requirements (obtained by BSR reported by users and the inspection of the internal buffers). On-demand data is instead requested for, e.g., the CPU consumption used in `vrAIn` or the Power used in `SBP-vRAN`, which could be retrieved in larger batches. These variables are not directly involved in the NI algorithm operation but are instead used to build accurate models of the vRAN system for the computing profile and power consumption.

In the considered NI examples, the Analyze blocks can be similarly shared. Driven by the same objective (i.e., the dimensionality reduction from a very informative yet large data), both `vrAIn` and `SBP-vRAN` build on an autoencoder-based analysis of the data, yielding succinct input data for the downstream learning agent. Here, the NI implementation can take advantage of Zenoh Flow feature of recursively defining each node: that is, each Analyze block can be composed of a

TABLE II: The N-MAPE-K definition for the `vrAIn` [11], and `SBP-vRAN` [12] algorithms.

Analytics	**Description**	
	`vrAIn`	`SBP-vRAN`
Sensors + Monitor	*Channel conditions*: SNR measurements, *traffic demands*: as Buffer State Reports (BSRs) from the terminals.	*Channel conditions*: SNR measurements, *Traffic demands:* BSRs and Downlink buffer occupation
Analyze	Inputs are passed through an autoencoders to reduce their dimensions, forming an encoding that is used in the execution algorithm.	Inputs are passed through an autoencoders to reduce their dimensions, forming an encoding that is used in the execution algorithm.
Plan	An actor-critic deep learning algorithm takes takes the encodings as input and generates two outputs: the amount of CPU required and the MCS policy.	A Bayesian Learning model takes the encodings as input and generates three outputs: MCS policy, airtime policy, TX power policy
Execution + Effector	Two APIs exposed by the virtualization environment (for the CPU quota) and the base station (for the MCS policy, via O-RAN A1/E2 interface).	Three APIs exposed by the base station (O-RAN E2 interface)
Knowledge	A *model* of the CPU behaviour of a base station.	A *model* of the power consumption behavior of a base station.
Training/Loss State/Actions/Rewards.	*states*: Latent representation of the input data; *actions*: compute and radio policies; *rewards*: latency tolerance.	*states*: Latent representation of the input data; *actions*: radio policies; *rewards*: maximum throughput subject to power budget.

set of Zenoh Flow nodes that, when joined together, provide the autoencoder functionality. Note that different outlets of each block can be placed at diverse positions yielding to, e.g., an autoencoder that compresses down to 4 or 8 dimensions according to the given use. These items can directly feed the Plan part, in charge of the actual decision in the system, or be stored in the knowledge blocks (e.g., for NI training).

The Plan blocks implement the trained learning agents for the two algorithms. They could have cascading relations (such as the CPU and Radio schedulers in `vrAIn`) or be completely independent, like the one used by `SBP-vRAN`. In principle, other relations may exist, such as peer-to-peer exchanges of information (e.g., for encodings) or achieved rewards computed by the Plan blocks.

Besides the data used for training, the knowledge nodes are fed with the policies and decisions computed by the Plan nodes, resulting in a database for the sink nodes in the network. Again, sinks can be shared across NIs instances, as in the case of the set MCS API, which is common to both `vrAIn` and `SBP-vRAN`. The end-to-end computation of decision (e.g., the Plan to Sink timing constraint, from epoch to epoch) can be implemented using the timestamping feature of Zenoh. For instance, the timing requirements of `vrAIn` and `SBP-vRAN` (i.e., 1s and 100ms, respectively) can be indicated to the Zenoh orchestrator, allowing to identify late inputs. Ultimately, the discussion above shows how, building on the proposed N-MAPE-K representation, the Zeno Flow framework can be used to implement heterogeneous NI algorithms.

VI. Conclusion

We presented evolutions in the representation and implementation of NI instances into operational mobile network architectures envisioned within the DAEMON project. We demonstrated with two practical AI/ML algorithms how N-MAPE-K-driven modeling eases an effective and synergic integration of NI in an open-source data flow programming framework. We believe that the approach proposed in this paper can be considered preparatory for introducing an intelligence layer in the network.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no.101017109 "DAEMON".

References

[1] I. Paez *et al.*, "DAEMON Deliverable 2.1: Initial report on requirements analysis and state-of-the-art frameworks and toolsets," Jun. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5060979

[2] A. Banchs *et al.*, "Network Intelligence in 6G: Challenges and Opportunities," ser. MobiArch '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 7–12.

[3] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez, and P. Rost, "Cares: Computation-aware scheduling in virtualized radio access networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 7993–8006, 2018.

[4] X. Ge, J. Yang, H. Gharavi, and Y. Sun, "Energy efficiency challenges of 5g small cell networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 184–191, 2017.

[5] Y. Wang, R. Forbes, C. Cavigioli, H. Wang, A. Gamelas, A. Wade, J. Strassner, S. Cai, and S. Liu, "Network management and orchestration using artificial intelligence: Overview of etsi eni," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 58–65, 2018.

[6] O-RAN Alliance, "AI/ML Workflow Description and Requirements v01.02.02," O-RAN Alliance, Technical Specification, 2020.

[7] ITU-T, " Architectural framework for machine learning in future networks including IMT-2020," ITU-T, Recommendation, 2019.

[8] ETSI, "Zero-touch network and Service Management (ZSM): Means of Automation," ETSI, Report, 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/005/01.01.01_60/gr_ZSM005v010101p.pdf

[9] O-RAN Alliance, "Architecture Description v02.00.00," O-RAN Alliance, Technical Specification, 2020.

[10] M. Camelo *et al.*, "Requirements and specifications for the orchestration of network intelligence in 6g," pp. 1–9, 2022.

[11] J. A. Ayala-Romero *et al.*, "vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[12] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Bayesian online learning for energy-aware resource orchestration in virtualized rans," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[13] S. Bhaumik *et al.*, "Cloudiq: A framework for processing base stations in a data center," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, 2012, pp. 125–136.

[14] J. A. Ayala-Romero *et al.*, "Orchestrating energy-efficient vrans: Bayesian learning and experimental results," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[15] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[16] W. M. Johnston, J. R. P. Hanna, and R. J. Millar, "Advances in dataflow programming languages," *ACM Comput. Surv.*, vol. 36, no. 1, pp. 1–34, 2004. [Online]. Available: https://doi.org/10.1145/1013208.1013209

[17] J. Dai *et al.*, "{HiTune}:{Dataflow-Based} performance analysis for big data cloud," in *USENIX ATC*, 2011.