M. Camelo et al., "Requirements and Specifications for the Orchestration of Network Intelligence in 6G," 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), 2022, pp. 1-9

DOI: 10.1109/CCNC49033.2022.9700729.

Requirements and Specifications for the Orchestration of Network Intelligence in 6G

Miguel Camelo, Luca Cominardi, Marco Gramaglia, Marco Fiore, Andres Garcia-Saavedra
Lidia Fuentes, Danny De Vleeschauwer, Paola Soto-Arenas, Nina Slamnik-Krijestorac, Joaquín Ballesteros,
Chia-Yu Chang, Gabriele Baldoni, Johann M. Marquez-Barja, Peter Hellinckx, and Steven Latré

***Abstract*—Next-generation mobile networks are expected to flaunt highly (if not fully) automated management. To achieve such a vision, Artificial Intelligence (AI) and Machine Learning (ML) techniques will be key enablers to craft the required intelligence for networking, i.e., Network Intelligence (NI), empowering myriad of orchestrators and controllers across network domains. In this paper, we elaborate on the DAEMON architectural model, which proposes introducing a NI Orchestration layer for the effective end-to-end coordination of NI instances deployed across the whole mobile network infrastructure. Specifically, we first outline requirements and specifications for NI design that stem from data management, control timescales, and network technology characteristics. Then, we build on such analysis to derive initial principles for the design of the NI Orchestration layer, focusing on (i) proposals for the interaction loop between NI instances and the NI Orchestrator, and (ii) a unified representation of NI algorithms based on an extended MAPE-K model. Our work contributes to the definition of the interfaces and operation of a NI Orchestration layer that foster a native integration of NI in mobile network architectures.**

***Index Terms*—Network Intelligence, Mobile Networks, Orchestration, 6G**

I. Introduction

Throughout the fifth generation (5G) and even in the forthcoming sixth generation (6G) era, mobile networks are being redesigned to support the extreme requirements set by future services that will assume performance indicators, such as virtually infinite capacity or perceived zero latency. Current standardization efforts reflect this trend by fostering end-to-end softwarization and cloudification of the network, which will ensure an unprecedented capacity to control system-wide operations [1]–[3]. Simultaneously, the classical access-core domain architecture of mobile networks is further atomized into *micro-domains* that substantially increase the management granularity of the infrastructure. An example of such micro-domains are the beyond and far edges introduced in [4], which decomposed the traditional access-edge domain [5] even more. To this end, edge computing is quickly becoming pivotal to enable such fine-grain management and to support latency-sensitive (e.g., vehicular communications) and high-bandwidth (e.g., virtual reality) use cases. Consequently, a wide spectrum of controllers, i.e., the element responsible to provide access to connectivity and physical network function resources within a domain, and orchestrators, i.e., the element responsible for deployment and dynamic optimisation of network services, are expected to operate and interact over the different micro-domains of the mobile network.

Future mobile networks abiding by the architectural model above will require advanced algorithms, using Artificial Intelligence (AI) and Machine Learning (ML) techniques, run by heterogeneous controllers and orchestrators to manage various micro-domains [6]. These algorithms can automatically manage the composite mosaic of network functions and associated resources consumed by a variety of network services (e.g., network slices) exploited by different tenants to craft the intelligence for networking, i.e., *Network Intelligence* (NI). More specifically, an NI instance is defined as a pipeline of effective AI/ML algorithms that swiftly detect or anticipate new requests or fluctuations in network activities, and then react to those by instantiating, relocating, or re-configuring virtual network functions in a fully automated manner.

Throughout the network, each controller and orchestrator shall run multiple NI instances to 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. These aspects are all highly critical, as the success and viability of 6G systems will largely depend on the quality and appropriate integration of NI solutions in the network infrastructure. Therefore, there is a need for revisiting the architectural design of mobile networks so that the operations of multiple NI instances can be accommodated across all micro-domains.

Present efforts promoted by major standardization bodies towards the integration of NI in next-generation network architectures pivot the notion of *closed-loop AI* [2]. According to this paradigm, the NI instances deployed at orchestrators and controllers operate in closed control loops. Abiding by the learning principles of closed-loop AI, such NI instances record the context of management decisions, collect observations about the quality of such decisions by continuing monitoring, and then use the feedback to improve future decision-making. A closed-loop model lets NI apprehend what is important to a stakeholder in a certain situation and learn over time to automate decision-making towards the targeting KPIs.

M. Camelo, P. Soto, N. Slamnik-Krijestorac, J. Marquez-Barja, P. Hellinckx, and S. Latré are with University of Antwerp - imec, IDLab. L. Cominardi and G. Baldoni are with ADLINK Technology. M. Gramaglia is with University Carlos III of Madrid. M. Fiore is with IMDEA Networks Institute. A. Garcia-Saavedra is with NEC Laboratories Europe. L. Fuentes and J. Ballesteros are with Universidad de Málaga. D. De Vleeschauwer and C.-Y. Chang are with Nokia Bell Labs

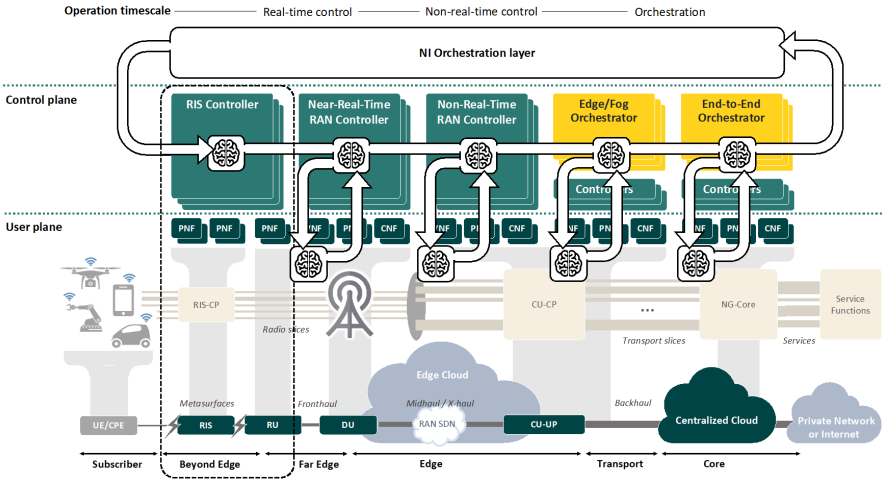

Fig. 1: The DAEMON NI-native architecture includes pervasive NI deployed within Network Functions (NFs) and user plane across different micro-domains, in which a novel beyond edge micro-domain is introduced to include Reconfigurable Intelligent Surfaces (RIS). NI instances thus operate at diverse timescales (at the top of the figure) and are coordinated by the NI Orchestrator in the overarching NI Orchestration layer to ensure efficient end-to-end decision-making.

The NI-native DAEMON architecture goes several steps beyond the current standardization trends [2], [6], [7] and posits a new approach for a systematic integration of NI in the 6G infrastructure, while staying fully aligned with emerging designs in standardization [4]. The concept underpinning the DAEMON model is illustrated in Figure 1. As outlined in the figure, this novel architecture allows for a much deeper embedding of intelligence in 6G systems and creates a considerably wider ecosystem of NI instances that populates the network infrastructure in both the control and user planes. This architecture allows breaking the centralized closed-loop model, by enabling hybrid models where user-plane components make their own decisions for 6G time-critical applications in disaggregated networks, e.g., L1/L2 beam management optimization in massive MIMO [8], which is not possible in fully centralized closed-loop models.

While the proposed architecture allows for very fast and local decision-making, the range of NI instances foreseen by the model needs to interact seamlessly to perform at their best, and exchange data and information to mutually improve both their learning and decision-making processes. To fully support a complex, pervasive and distributed NI-native environment, an *NI Orchestration layer* is introduced in Figure 1, a system to orchestrate NI instances that can be deployed on the network, which decision-making engine is the *NI Orchestrator*. This novel layer is responsible for supervising intelligence in the DAEMON architecture as a whole, ensuring the ideal functioning of each closed-loop NI instance, and overseeing interactions across closed loops that run NI at different timescales and different micro-domains. The two main NI management tasks for the NI Orchestration layer are summarized as follows:

- Selecting the best NI algorithm within each NI instance from a predefined set of algorithms. The variants of NI algorithms will be designed by employing different modeling strategies and adaptive learning techniques. Decisions on the most appropriate algorithm are based on contextual information (e.g., the reliability level of the traffic demand predictions), available system resources (e.g., the computational capacity that can be dedicated to the NI instance), performance requirements (e.g., the target precision of the algorithm), and the amount of data to be processed (e.g., the look-back for a forecasting algorithm), concurrently across timescale levels.
- Coordinating all NI instances in the system to ensure grateful operation of all live mechanisms operating at different timescales and in different micro-domains. To this end, the NI orchestration layer will support the exchange of information via dedicated interfaces with the NI instances and centrally the NI Orchestrator will solve trade-offs that may emerge from conflicting objectives in the control and user planes, e.g., in establishing policies (at small timescales) versus enforcing such policies (at large timescales).

In this paper, we present a set of requirements, specifications, and workflow for closed-loop control that are required to realize the envisioned NI orchestration layer. To this end, in Section II, we first present the set of requirements imposed by data management and control timescales for NI. We then outline the initial specifications for the NI orchestration layer which stem from infrastructure aspects of mobile networks, in Section III. Jointly considering these two sets of requirements and specifications is key to designing the NI orchestration layer, which can help NI algorithms overcome the intrinsic limitations of their local view (e.g., NI running in the user plane) and close to the optimal point of operation without sacrificing reactivity. Based on these considerations, we present a workflow that supports the design and deployment

of NI instances in end-to-end closed-loop control settings, in Section IV, which is needed to ensure system-wide stability without human intervention. Finally, conclusions and future work are discussed in Section V.

II. Data and Control Requirements for NI

A key step towards the design of a 6G network management model is to analyze the challenges and requirements imposed on applications and network functions in a distributed edge-to-cloud environment, especially when a fine-grain control loop is required with no human intervention. To that end, the items of particular concern are (1) the distribution and management of data among disaggregated infrastructure and (2) the impact of operating at different timescales for control systems. We analyze these two aspects in detail below.

A. Data distribution and management for NI

Edge infrastructures typically encompass a heterogeneous mix of resources that can be extremely diverse in terms of capabilities and operating conditions. In contrast, in most cases, cloud infrastructures are highly homogeneous. Nevertheless, it would be naive to consider the cloud and edge as two discrete locations. Instead, they are the two extremities of a continuum inside which computing, storage, and networking resources can be deployed at any point and in any topology [9]. This seamless configuration leads to resources that are highly heterogeneous in terms of (i) computing power (consider, e.g., a datacenter server compared to an end-user device or an embedded sensor), (ii) energy requirements (e.g., in the presence of battery-powered devices), (iii) mobility patterns (e.g., for on-board units embedded in cars or trains), (iv) traffic patterns (e.g., sensors producing large amounts of push-up data toward the network versus servers providing push-down data or video streams), and (v) activity patterns (e.g., always on versus long sleep mode devices).

Despite their diversity, resources are expected to collaborate to a certain degree to compose a unified infrastructure on which network functions and end-user applications can be deployed. This leads to a scenario which multiple application/system components (e.g., microservices) can run at very different locations within the network infrastructure, from the far edge all the way up to the cloud. In this case, one fundamental problem is how to effectively enable end-to-end data semantics in the entire application/system while easing the development, deployment, and management of the target application or network function, which can be either infrastructure-oriented or user-oriented. This is a fundamental requirement to truly enable an application or network function to be executed and migrated everywhere in the network without incurring heavy reconfiguration. From that point of view, we can summarize the main data patterns that an application or network function may experience:

- **Data are pushed or pulled**: This kind of pattern is very common in reactive systems, such as cyber-physical applications (e.g., robotics) or event-driven software paradigms (e.g., cloud-based telemetry). According to this pattern, an application/network function (i.e., publisher) decides when to push data into the system, which is then delivered to other applications/network functions (i.e., subscribers).
- **Data are computed on demand**: In this case, fresh data can be requested to applications/network functions, whenever deemed necessary.
- **Data are stored**: Some data that has been published in the system requires to be stored temporarily or permanently for later processing.
- **Data have different representations**: The format of the stored data may be different from the format they were originally produced. In other words, the format used when producing data may differ from that used when consuming the data.

As of today, a complex error-prone patchwork design is required to achieve end-to-end data semantics integrating the above patterns across the whole infrastructure, from tiny sensors up to powerful data centers [10]. Various network protocols must be stitched together to provide the necessary support, with an inherent burden on application complexity, system management, and troubleshooting. Assuming that an operator/application developer is willing to embrace the complexity above to achieve end-to-end semantics, there is a risk that the attempt will hinder the overall dynamic system optimization, and the approach imposes hard limits at the boundaries of each network tier.

To better frame the above in the NI-native DAEMON architecture, we can start from the observation that data produced in the network can be used for different purposes, such as on-the-wire transmission optimization, traffic and capacity forecasting, anomaly detection, proactive replicas for reliability, and application migration to meet traffic demands, among others. In all these cases, NI models are interested in processing some data (on-line or off-line) and producing actions as a result. These models are seldom interested in retrieving and maintaining data (e.g., where data are stored), or in what data format is used when producing them. Their only interest is in consuming and processing such data without dealing with the whole network complexity laying underneath. The NI-native architecture should hence provide a decentralized and unified data management approach to facilitate the development, operation and management of NI model. In particular, the envisaged requirements include:

- **Unification** of various data patterns in a single and harmonized platform supporting data in motion (i.e., publish/subscribe), data in use (i.e., caching), data at rest (i.e., storage), and on-demand computation (i.e., remote procedure calls).
- **Decentralization** by design to support the very distributed nature of the edge computing where resources and applications are expected to be more mobile, and volatile compared to traditional cloud computing.
- **Heterogeneity** by design to account for the great diversity of devices, resources, applications, and traffic/mobility

patterns that are expected to coexist in the edge infrastructure.
- **Consistency and Stability** to be more resilient to failures in the network and to guarantee that the overall system is not brought to a halt in case of temporary inconsistencies (e.g., due to network partitioning).
- **Wire efficiency** to reduce the overhead in the network, hence the overall energy consumption of the system.
- **Data pipeline** to support the staged processing of some data where multiple NI models may be chained together.

An example of a framework that can delivery a decentralized and unified data management platform is Zenoh[1], which is a novel data-centric protocol that allows sharing data in a location-transparent manner while supporting the data patterns above. In DAEMON, Zenoh is being consider as a candidate for the multi-timescale closed-loop NI framework to support end-to-end data semantics for the NI models [10].

The data are the main input/output for the development, operation, and management of any NI model. A single NI instance may comprise several components that operate at different locations in the network to accomplish an overarching goal. In the case of a general NI algorithm, we can identify two main operation modes: NI control and NI training. The former relates to the capability of running a trained model into the network to control it at different timescales according to its intended purpose (e.g., resource scheduling, link optimization, and capacity forecasting). The latter relates to the capability of consuming data and producing a model as output to be deployed later in the network for control reasons. As in this work we are interested in network management rather than algorithm training, our focus is on NI control, which is discussed next.

B. Control of NI at different operation timescales

In 6G, the automation of the orchestration and control of resources of access and edge infrastructure will play a crucial role. Decisions are being made on multiple layers.

1) The end-to-end orchestrator needs to make decisions (e.g., where to place or migrate virtual network functions) related to end-user information and end-to-end performance without having a detailed view on the status from individual domains.
2) The domain orchestrators and controllers take local decisions for the domain they control (e.g., life-cycle management of domain functions, allocating resource for processes, routing traffic), both taking the interaction with adaptive applications into account.

Note that in such multi-layer control, terminating the control loops may lead to instability if the control loops are not designed adequately. However, current decisions are taken prudently (and often manually) to avoid instability issues. Moreover, the control loops should be stable under any changing condition (e.g., the introduction of a new software release). Our goal is not necessary to improve the performance of the algorithms but rather the level of automation. Hence, a reasonably good enough algorithm that is likely to be more stable is preferred over an algorithm that is over-engineered for only a few specific situations.

The timescale of decisions taken by the end-to-end orchestrator is between typical inter-arrival times of network services (i.e., days, hours) and the typical duration of elevated traffic load produced by users of the network service (which fluctuations might impact the decision). The timescale at which the domain controllers need to make decisions is much shorter (seconds to sub-seconds). As explained above, the end-to-end orchestrator will only have a limited view of the details in the domain. Similarly, all the desired data upon which the controller needs to make decisions may not be available in a timely manner. Therefore, domain controllers will also have to take decisions based on partial observations. One of the research questions to be tackled is to investigate how the status of the domains can be summarized concisely, so that the orchestrator can still make beneficial use of it to make sensible decisions without requiring huge information exchange, and to isolate sensitive data in the domain for security issues.

Also, it is worth mentioning that such different timescales between orchestrators and local controllers will limit the design space of NI algorithms. For instance, the real-time controller can make local decisions over their working time interval (e.g., milliseconds, seconds) only based on the resource provisioned by the orchestrator; in contrast, the orchestrator monitors the performance statistics of individual controllers and makes the scaling and resource provisioning decisions on a larger time interval (e.g., hours, days). Such hierarchical decision-making mechanism is beneficial to craft independent decision-making algorithms; nevertheless, it may pose significant limits on the optimization goals, once such hierarchy under-utilize or over-utilize the information from the others. Therefore, another research question is how to craft such multi-layer control loop in a reasonable hierarchy with few extra constraints and small performance bound, compared with the non-hierarchical one.

Orthogonal but heavily intertwined with intra-network service control and orchestration is the dynamic scheduling of (typically scarce) shared physical resources among them (inter-network services). Based on priority or relative weighted policies, resources need to be assigned to different network services with different objectives. Therefore, the architecture of control and orchestration should support a composition of multi-layer hierarchical intra- and inter-network service control and orchestration, each level driven by high-level intent objectives, e.g., high-level QoE targets and business KPIs.

Finally, the aforementioned high-level per-network service intent itself should be capable of being compared with each other to facilitate the decision-making by the controller/orchestrator, under the provisioned comparison rules. These rules can be viewed as the inter-network service intent, which can be stateless or stateful. To be more specific, the stateless rules do the intent comparison without considering the per-network service current status (e.g., execution time,

[1]Zenoh project. Online: https://zenoh.io. [Accessed 08 November 2021]

statistics, active end-users), whereas the stateful rules take the network service status into consideration and make the comparison correspondingly. To sum up, different intent comparison mechanisms should be considered in the architecture.

For what concerns NI control, in the following we focus on the timescale at which data are produced, stored and consumed by considering the following classes.

- **Very small timescale (us-ms)**: This refers to the case where a constant stream of data is produced and consumed locally for taking localized optimization decisions. At this rate, it's not feasible to store the data produced for a long period of time (e.g., minutes or hours), given the large amount of space required. However, according to recent history, storing particular data in a small cache can be used to optimize the local system. The system is mainly characterized by high-throughput data producers and consumers, and, for performance reasons, data producer, consumer and storage need to be collocated. The task to be accomplished is very specialized, leading to a well-defined dataset and optimization procedures. The NI models target exclusively on-line optimization. A data pipeline may be used to run multiple independent NI models in parallel for redundant decisions.
- **Small timescale (ms-s)**: This is similar to the previous case, with the exception that data producer, consumer and storage may be distributed. The dataset and type of optimization procedures are still domain-particular and specialized. However, at this timescale, they can take the flavor of a local collaborative optimization with the interaction between neighboring components. The NI models target exclusively on-line optimization. A data pipeline may be used to chain NI models running on different nodes for collaborative decisions.
- **Medium timescale (s-min)**: More generic optimizations can be run in a limited portion of the overall system for this case. Datasets are more heterogeneous where multiple states of the local system can be considered together (e.g., network load, traffic pattern, UE mobility) to perform reactive/proactive optimization of the mobile network. In this case, a balanced mix of data producers, consumers, and storage is considered to quickly react to events in the network and operate in response to the recent history of the system. Multiple NI models can be chained together in a decentralized data pipeline where data are processed locally but decisions are taken at the system level. The NI models target both on-line and (limited) off-line optimization [11]. A data pipeline may integrate fresh and historical data for optimization.
- **Large timescale (min-h)**: Optimizations at the complete system level is feasible from this class. Datasets are heterogeneous and composed of aggregated data. In this case, data storage is the main source of data, whereas data producers serve the purpose of updating the storage with aggregated data. A decentralized data pipeline can serve the purpose of aggregating the data first and processing them next. NI models for intelligently aggregating data will need to work in an on-line fashion while NI models performing system-level optimization are expected to work in an off-line fashion. A data pipeline may be used to perform a multi-stage optimization.
- **Very large timescale (h-days)**: At this timescale, only system-level optimization is performed. Data storage is the only source of data, and both dataset and optimization procedure types are extremely heterogeneous. The optimization is performed purely off-line. A data pipeline may include a human in its loop.

TABLE I: Summary of NI control timescale characteristics.

NI control timescale	Time units	Optimization level	On-line/Off-line optimization
Very short	Microseconds-Milliseconds	Local	On-line
Short	Milliseconds-seconds	Local/Collaborative	On-line
Medium	Seconds-minutes	Local	On-line and off-line
Long	Minutes-hours	System level	Off-line
Very long	Hours-days	System level	Off-line

Table I presents a summary of the characteristics of different timescales. An additional important aspect related to control timescales is the capability of the control system to operate in a timely manner, guaranteeing a response within a specified timing constraint. From a system-level perspective, we can say that the control system needs to operate in real-time according to the corresponding timescale(s). It is important to remark that in here *we refer to real-time not as a synonym of low latency but rather to express the capability of the control system to meet a given deadline at any timescale of interest.*

III. Network-driven specifications for NI

Network-driven specifications of NI design are especially linked to the time dimension: aspects such as data collection time or data processing time depend on the underlying infrastructure and its capability, or directly on the time needed to execute the NI algorithms, yielding an overall higher or lower time required to control the network. In Figure 2, we present a schematic view of the different scenarios that may emerge when dealing with NI in production mobile networks.

The **infrastructure monitoring data** consists of all the input data collected and provided by the many and varied measurement probes deployed in the infrastructure, at any micro-domain (e.g., core, transport, edge, far-edge). This could come from the network functions themselves, from the infrastructure hosting them, or from the transport network interconnecting them. More specifically, the monitoring dimension could be further split into three sub-dimensions, which offer different trade-offs, as follows.

- **Data sampling rate** is the frequency at which data is gathered. Higher sampling rate force the infrastructure to have more capacity to support his.
- **Dataset richness** is the number of details associated to the data. For instance, a base station can report averaged values for the Signal to Noise ratio or directly, per User Equipment (UE) values. This includes the data granularity and the dimension of each item.

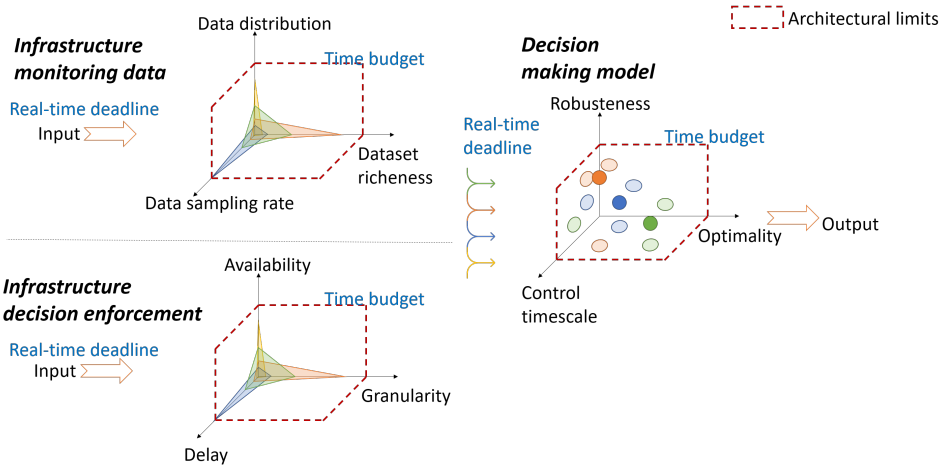

Fig. 2: Network Intelligence degrees of freedom, input-output relationship, and real-time constraints subject to infrastructure observability and controllability.

- **Data distribution** identifies the sampling location of the network or from the entire network. Similar KPIs can be gathered from different functions, e.g., collect user throughput from both radio access and core network together or just from core network.

The **infrastructure decision enforcement** includes all the parameters that can be changed, reconfigured and orchestrated, typically in a programmable way through Application Programming Interfaces (APIs). Analogously to the monitoring data, the enforcement may happen at any level in the network. Again, this dimension can be split into three sub-dimensions, according to their trade-offs:

- **Delay** is the time it takes to generate the output of the algorithm and inject it through the API into the specific asset (e.g., network function or infrastructure). This time is consumed by the budget available for the action.
- **Availability** means that an action may be enforced at every opportunity, or it may be resilient to some missed opportunities. For instance, scheduling patterns may be enforced at every Transmission Time Interval (TTI) or may be performed at a coarser time interval.
- **Granularity** indicates the scope of the application of an action. For instance, the highest level of Modulation and Coding Scheme (MCS) could be applied at a single base station or specified for every single UE.

Given the time constraint and the underlying infrastructure capabilities, there is a finite set of feasible combinations that match all the requirements specified before.For instance, an algorithm may gather data at the UE level, but , if not served by a subset of the same base stations, its actions cannot be enforced at the UE level. Or, if the algorithm requires 1ms data sampling rate and 1ms-bounded action enforcement, then the gathered metrics and configurations that can be applied to the network infrastructure can only show good performance when applied to the same network function.

Once the timing constraints are in place, according to the specific network problem to be solved, a set of NI solutions is selected and implemented. However, there is a further degree of variability to be considered, which is intrinsic to the selected **model/algorithm for decision-making**. Again, this can be articulated into three sub-dimensions:

- **Robustness** indicates the resiliency of the NI solution to missed inputs. For instance, a suitably trained Neural Network (NN) may be capable of extrapolating the output from past inputs, with a likely higher degree of uncertainty, but still providing a reliable outcome. Instead, an optimization algorithm without any memory may be very unreliable or directly inapplicable when inputs are missing due to, e.g., delays in the monitoring infrastructure.
- **Optimality** means the robustness and the precision that data driven solutions, such as the one based on NNs, may achieve at a price of a higher resource utilization (complex models are fast mostly when used on a GPU), and a certain degree of obscurity in their decision. Therefore, for some problems, the trade-off between the optimality of the decision and the time required to take that decision should be considered, for example, using a simpler model or directly using traditional optimization algorithms.
- **Control timescale** is similar to the availability of infrastructure decision enforcement, which poses an upper bound to this value. NI algorithms can then decide to exploit all configuration opportunities or not.

Similar to the previous dimensions, these three sub-dimensions pose a trade-off surface that needs to be addressed, especially by the NI Orchestration layer proposed by DAEMON. A fundamental factor here is the dichotomy between AI and ML models and the ones based on traditional mathematical optimization, that stay at the different ends of the spectrum: faster to obtain good approximation results but potentially more fragile the former, optimal but slow the latter [12], [13].

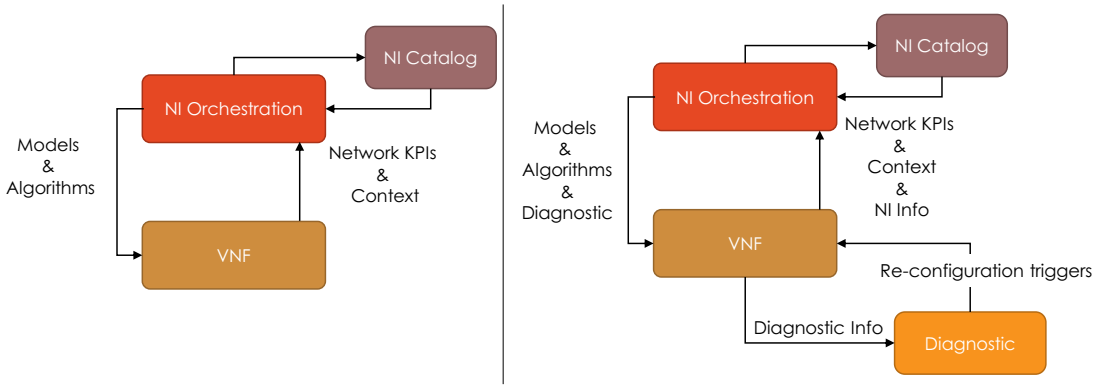

Fig. 3: Two strategies for control loops between NI instances and NI Orchestration layer.

IV. Design principles of NI Orchestration layer

Based on the requirements and specifications drafted in Sections II and III, we finally propose some initial guidelines for the design of a closed control-loop of NI instances via the NI Orchestration layer, such as the one proposed by the DAEMON architectural model in Figure 1. Specifically, we first envision two strategies to implement the control loops between individual NI instances and the NI Orchestration layer, in Section IV-A. Then, we discuss how to represent the operation of individual NI instances in a unified way, in Section IV-B, which paves the road to the definition of a NI-agnostic interface between the NI instances and the NI Orchestration layer.

A. Looping NI instances and NI Orchestration layer

NI models are trained with data gathered from the network, and they possibly keep learning on-line (cf. TABLE I), during the network operation, through reinforcement learning techniques. However, due to the very fast reaction times, model training and updates at the network edge cannot be performed at wire speed, for a number of reasons, including the possibly long training times and the lack of availability of specialized hardware such as GPUs. Thus, the environment shall include modules and interfaces that allow the NI Orchestration layer, as discussed previously, to continuously monitor and possibly update the NI modules running at the edge. So far, we identified two possible workflow strategies for this task, as depicted in Figure 3.

The first one, depicted on the left part of Figure 3, describes bi-directional interaction between NI Orchestration layer and NI instance in the form of network function:

- From NI Orchestration layer to the Network Function: NI orchestrator picks the best NI algorithms to perform specific tasks from a NI Catalog, as shown in Fig. 3 right.
- From the Network Function to NI Orchestration layer: Network function reports on its achieved KPI (e.g., KPI targets listed in Section I) and some information about the context (i.e., the network conditions observed at the moment). This information allows the NI orchestrator to monitor the KPIs and understand whether the context associated to the NI is still the one originally used for the training.

However, this could be limiting for some use cases that require a tighter interaction between the "drift" control and its actual implementation. Thus, in the second approach (depicted on the right-hand side of Figure 3), NI orchestration sends an additional diagnostic module that can perform these computations on the drift very close to the network functions, attempting to maximize the performance provided by NI instance at any point in time.

B. Unified representation of NI algorithms

In order to fully integrate NI algorithms into the overall architecture, it is fundamental to understand what are the needed interfaces that algorithms use to interact with their environment. For this purpose, we adopted 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 model for autonomic and self-adaptive systems [16].

In a nutshell, the MAPE-K specifies how the different modules have to interact with the system, as follows:

- The **Sensors** specify all the probes that are needed to gather the input data and the kind of input data we have to gather. In principle, the APIs are specified at this level.
- The **Analyze** block includes any pre-processing, summary, or preparation of the data such as averages, auto-encoders, clustering algorithms.
- The **Plan** implies the specific NI algorithm that is implemented, for instance a Neural Network (NN) fulfilling a categorization tasks.
- The **Execute** part specifies how the algorithm is going to interact with the system and possibly change configuration parameters.
- Finally, the **Effector** includes specific configuration parameters updated in the Network Function, again specifying the API.

The algorithms that run at NI instances can be classified in a unified manner, according to how they interact with the other elements of the network.

TABLE II: The MAPE-K definition for the vrAIn [14] and ATARI [15] algorithms.

Analytics	Description	
	vrAIn	**ATARI**
Sensors + Monitor	Channel Conditions, SNR measurements, traffic demands (as Buffer State reports from the terminals).	Link state (SINR, RSSI), AP's available channel configurations, distance and associations among users and APs, users' traffic demands
Analyze	Inputs are passed through an autoencoders to reduce their dimensions, forming an encoding that is used in the execution algorithm.	Inputs are converted in a graph-like structure to capture all the topological properties of the WLAN deployent.
Plan	An actor-critic deep learning algorithm, that takes the encodings as input and generates two outputs: the amount of CPU required and the maximum MCS.	A GNN to predict the throughput at each client and AP.
Execution + Effector	Two APIs exposed by the virtualization environment (for the CPU quota) and the base station (for the maximum MCS).	The predictor can be part of a digital twin and be used by a WLAN controller to solve the WLAN channel bonding optimization problem.
Knowledge	A model of the CPU behaviour for the different decoding tasks.	Monitored data, current setup, best performance prediction, possible channel configurations, learned channel configuration selection policy.
Training/Loss State/Actions/Rewards.	states: Latent representation of the input data, actions: compute and radio control, rewards: maximum latency.	Model: GNN, Loss Function: ot Mean-Squared Error as loss function.

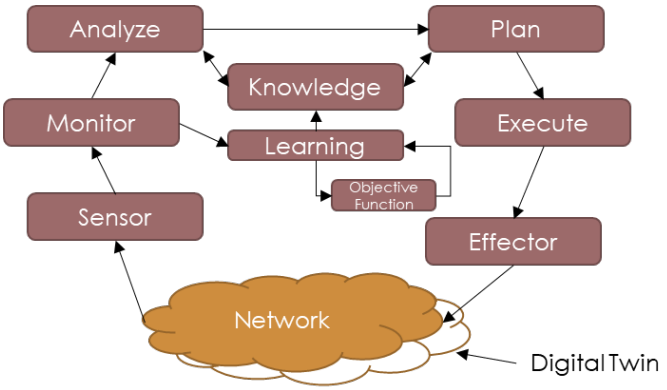

Fig. 4: Extended MAPE-K abstractions for NI algorithms.

However, 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 our environment, as depicted in Figure 4. Specifically, two modules shall be added to the plain MAPE-K depending on whether the NI algorithm is based on supervised, unsupervised, or reinforcement learning. The two modules are the **Learning** and the **Objective Function** to optimize via learning, which will connect and enhance the knowledge module. The learning module specifies aspects such as the input data shape, batches, and most importantly, the algorithm used to learn how to solve a given task. In supervised learning, the learning algorithm can be a NN, a decisions tree, etc. The objective function is the **loss function** associated with the task to be solved, e.g., regression or classification.

As for unsupervised learning, the learning algorithm can be K-means, Gaussian Mixtures, NN (e.g., in the form of AutoEncoders), and the objective function is a loss function (or cost function) that characterizes what the data looks like, e.g., the minimum distance squared to the representatives in a cluster. Finally, when applied to reinforcement learning, the learning module and objective function are replaced with a **State/Action** representation and a **reward function**, respectively. Additionally, the effector and the sensors can also be redirected to a **Digital twin** element, if needed by the specific NI instance. This digital twin, in the form of simulators, emulators, test environments, etc., will play a key role in bridging the gap between NI algorithm and communication systems, so it is possible to train, test, and validate NI instances before being applied to the operating network [17].

In Table II, we provide the extended MAPE-K definition for the vrAIn [14] and ATARI [15] algorithms as examples of NI. *vrAIn* is a reinforcement learning algorithm that tailors the available computing capacity on a vRAN platform to the expected load introduced by the different PHY layer tasks such as frame decoding. Thus, by gathering information from the network operation (such as the link conditions and the load introduced by the different terminals) it can compute a suitable policy for the amount of computing resources assigned and a cap on the maximum modulation and coding scheme (which limits the stress on the computing platform). *ATARI* is a Graph Neural Network (GNN) model that exploits the information carried in the deployment of WLANs to predict its performance with high accuracy. The idea behind this predictor is to provide an approach that combines the best of the two more common techniques for this task: faster than state-of-the-art and high-accurate simulators and minimal drop in accuracy compared to simpler analytical models or other data-driven state-of-the-art ML approaches (e.g., CNN). This model is very suitable as a digital twin for WLAN controllers that want to evaluate the impact of different configurations of spectrum channels among access points and select the one that maximizes the performance of the deployment.

Ultimately, using the proposed approach allows specifying different kinds of input data, different ways of training a NN (supervised learning or reinforcement learning) and a reconfiguration trigger for the network function. By summarizing the algorithms in this way, it will be possible to devise the kind of interfaces that will be needed for the NI-native architecture.

V. Conclusions and Future Work

In this paper, we elaborated on the DAEMON architectural model [4], which proposes introducing an NI orchestration layer for the effective end-to-end coordination of NI instances deployed across the mobile network infrastructure for 6G. And we outlined requirements and specifications for NI design that stem from data management, control timescales, or network technology characteristics. Based on the analysis of such

requirements and specifications, we derived initial principles for the design of the NI orchestration layer, focusing on (i) proposals for the interaction loop between NI instances and the NI orchestrator, and (ii) a unified representation of NI algorithms based on an extended MAPE-K model. With this approach, we provided a mechanism to define the NI orchestration layer's interfaces and operations that foster a native integration of NI in mobile network architectures.

As future work, we will continue to evolve the NI orchestration layer and investigate several aforementioned research questions. In specific, we expect to investigate how the micro-domain status can be summarized concisely and evaluate the multi-layer control loop approach, as mentioned in Section II. Additionally, a further deep dive into our extended MAPE-K model with other NI instances is planned as another potential direction. We plan to map our MAPE-K framework to current standardization frameworks that include the interaction of traditional network architectures with AI/ML, such as ETSI ENI or ZSM [2], [7], and possibly to contribute to further development of such efforts. Finally, further research on security aspects is required on the extended MAPE-K model, for example by introducing a MAPE-SAC (security assurance cases), which is a security-focused feedback control loop, and its interaction with the MAPE-K, which is a performance-focused control loop.

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] D. Bega, M. Gramaglia, R. Perez, M. Fiore, A. Banchs, and X. Costa-Pérez, "Ai-based autonomous control, management, and orchestration in 5g: From standards to algorithms," *IEEE Network*, vol. 34, no. 6, pp. 14–20, 2020.

[2] 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.

[3] A. Garcia-Saavedra and X. Costa-Perez, "O-ran: Disrupting the virtualized ran ecosystem," *IEEE Communications Standards Magazine*, pp. 1–8, 2021.

[4] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia, "Network intelligence in 6g: Challenges and opportunities," in *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*, ser. MobiArch '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 7–12. [Online]. Available: https://doi.org/10.1145/3477091.3482761

[5] ETSI, "Autonomous Networks, supporting tomorrow's ICT business," ETSI, White Paper, 2020. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi-wp-40-Autonomous-networks.pdf

[6] ITU-T, "ITU-T Y.3172-series - Architectural framework for machine learning in future networks including IMT-2020," ITU-T, Recommendation, 2019. [Online]. Available: http://handle.itu.int/11.1002/1000/13894

[7] 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

[8] O-RAN, "O-RAN Use Cases and Deployment Scenarios," O-RAN, White Paper, 2020. [Online]. Available: https://static1.squarespace.com/static/5ad774cce74940d7115044b0/t/5e95a0a306c6ab2d1cbca4d3/1586864301196/O-RAN+Use+Cases+and+Deployment+Scenarios+Whitepaper+February+2020.pdf

[9] E. Foundation. (2021) From devops to edgeops: A vision for edge computing, white paper. [Online]. Available: ttps://outreach.eclipse.foundation/edge-computing-edgeops-white-paper

[10] G. Baldoni, J. Loudet, L. Cominardi, A. Corsaro, and Y. He, "Facilitating distributed data-flow programming with eclipse zenoh: The erdos case," in *Proceedings of the 1st Workshop on Serverless Mobile Networking for 6G Communications*, ser. MobileServerless'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 13–18. [Online]. Available: https://doi.org/10.1145/3469263.3469858

[11] A. D. Filippo, M. Lombardi, and M. Milano, "The blind men and the elephant: Integrated offline/online optimization under uncertainty," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 4840–4846, survey track.

[12] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221720306895

[13] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 16–25. [Online]. Available: https://doi.org/10.1145/1128817.1128824

[14] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "vrain: Deep learning based orchestration for computing and radio resources in vrans," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[15] P. Soto, M. Camelo, K. Mets, F. Wilhelmi, D. Góez, L. A. Fletscher, N. Gaviria, P. Hellinckx, J. F. Botero, and S. Latré, "Atari: A graph convolutional neural network approach for performance prediction in next-generation wlans," *Sensors*, vol. 21, no. 13, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/13/4321

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

[17] F. Wilhelmi, M. Carrascosa, C. Cano, A. Jonsson, V. Ram, and B. Bellalta, "Usage of network simulators in machine-learning-assisted 5g/6g networks," *IEEE Wireless Communications*, vol. 28, no. 1, pp. 160–166, 2021.