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. Network Intelligence (NI) will be the key enabler for such a vision, 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 and beyond their fifth generation (B5G), mobile networks are being redesigned to support the extreme requirements set by future services that will assume performance indicators like virtually infinite capacity or perceived zero latency. Current efforts in standardization reflect this trend by fostering the end-to-end softwarization and cloudification of the network, which will ensure unprecedented capacities to control system-wide operations [1]–[3]. At the same time, the classical access-core architecture is being atomized to create network micro-domains that substantially increase the management granularity of physical infrastructures. To that end, Edge computing is quickly becoming pivotal to enable such fine-grain management as well as to support latency-sensitive (e.g., vehicular communications) and high-bandwidth (e.g., virtual reality) use cases. As a result, a wide spectrum of controllers and orchestrators are expected to operate and interact in the Core, Transport, Edge, and Far Edge micro-domains of the mobile network architecture.

Network Intelligence and standardization. Future mobile networks abiding by the architectural model above will require that advanced algorithms are run by many heterogeneous controllers and orchestrators. Such algorithms shall be capable of automatically managing the composite mosaic of network functions and associated resources consumed by a large number of network services (e.g., network slices) owned and exploited by various tenants. Here, *Network Intelligence* (NI) will play a fundamental role. A NI instance is a pipeline of effective algorithms that swiftly detect or anticipate new requests or fluctuations in the 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 number of Key Performance Indicator (KPI) targets; these include Quality of Service (QoS) or Quality of Experience (QoE) guarantees, maximization of infrastructure and resource reuse across multiple tenants or network services, and full network automation to achieve zero-touch network and service management. These aspects are all highly critical, to the point that the success and viability of B5G systems will largely depend on the quality and appropriate integration of NI solutions in the network infrastructure. There is therefore a need for revisiting the architectural design of mobile networks so that it can best accommodate the operation of NI across all micro-domains.

Present efforts promoted by major standardization bodies towards the integration of NI in next-generation network architectures pivot on the notion of *closed-loop Artificial Intelligence* (AI) [2]. According to this paradigm, the NI instances deployed at centralized orchestrators and controllers work in closed control loops. Abiding by the learning principles of modern AI, such NI instances record the context of management decisions, collect observations about the quality of such decisions via continuing monitoring, and then use the feedback to improve future choices. A closed-loop model lets NI apprehend what is important for an operator in a certain situation and learn over time to automate optimal decision making towards the expected KPI targets listed before.

Towards NI-native mobile networks. The DAEMON architecture is a NI-native B5G architectural model that goes several steps beyond the current standardization trends, and posits a new approach for a more systematic integration of NI in the B5G 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

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. Chang are with Nokia Bell Labs

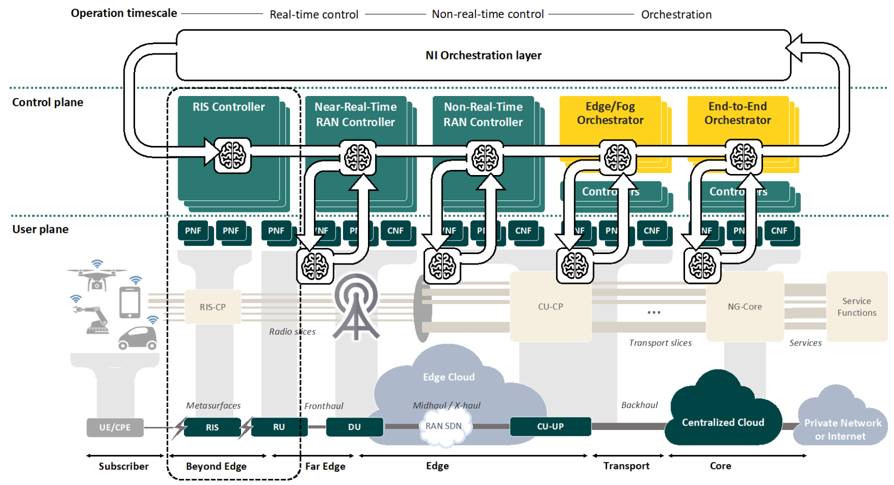

Fig. 1: Concept of the DAEMON NI-native architecture. Pervasive NI is deployed not only across all mobile network micro-domains (including a new *Beyond Edge* that includes, e.g., Reconfigurable Intelligent Surfaces) but also within Network Functions (NFs) and possibly directly in the user plane. NI instances thus operate at very diverse timescales (on top of the figure), and are coordinated by an overarching NI Orchestration layer that ensures efficient end-to-end decisions.

deeper embedding of intelligence in B5G systems and creates a considerably wider ecosystem of NI instances that populate the network infrastructure both in 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 time critical operations, which is not possible in strongly centralized closed-loop models.

While the DAEMON approach allows for very fast and localized decision-making, the range of NI instances foreseen by the model need to interact seamlessly to perform at their best, and exchange data and information so as to mutually improve both their learning and decision-making processes. In order to provide a network architecture that fully supports the complex, pervasive and distributed NI environment, the DAEMON model also introduces a NI Orchestration layer as per Figure 1. This novel layer is responsible for supervising intelligence in the network 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. The two main NI management tasks for the NI Orchestration layer can be 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 modelling strategies and adaptive learning techniques. Decisions on the most appropriate algorithm will be 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 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 Orchestrator will support the exchange of information via dedicated interfaces with the NI instances, and centrally solve trade-offs that may emerge from conflicting objectives in the control and data planes, e.g., in establishing policies (at slow timescales) versus enforcing such policies (at fast timescales).

Designing NI orchestration. In this paper, we present a set of requirements, specifications, and the workflow for a closed-loop control that are required to realize the NI Orchestration layer envisioned in [4]. To this end, we first present the set of requirements imposed by data management and control timescales for NI, in Section II. We then outline the initial specifications for the NI Orchestrator that stem from infrastructure aspects of mobile networks, in Section III. Jointly considering these two sets of requirements and specifications is key to design a NI Orchestrator that can help NI algorithms overcome the intrinsic limitations of their local view (e.g., NI running in the data plane) and close in on the optimal point of operation without sacrificing reactivity. Based on these considerations, we present a workflow that supports designing and deploying NI instances in an end-to-end closed-loop control settings, in Section IV, which is needed to ensure system-wide stability without human intervention. Conclusions and future work are finally discussed in Section V.

II. Data and Control Requirements for NI

A key step towards the design of a B5G network management model lies in analyzing the challenges and requirement imposed to applications and network functions by the distributed edge-to-cloud environment, especially when a fine-grain control loop is required with no human intervention. To that end, items of particular concern are: (1) the distribution and management of data in such a scattered infrastructure; and, (2) the impact of operating at different timescales for control systems. We analyze these two aspects in detail next.

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, cloud infrastructures are stationary and highly homogeneous. Nevertheless, it would be naive to consider the cloud and edge as two discrete locations. Rather, they are the two extremities of a continuum inside which compute, storage, and networking resources can be deployed at any point and in any topology [5]. This seamless configuration leads to resources that are highly heterogeneous in terms of: (i) computing power (consider, e.g., a datacenter server compared to a end-user device or an embedded sensor); (ii) energy requirements (e.g., in presence of devices that are battery-powered); (iii) mobility patterns (e.g., for on-board units embedded in cars or trains); (iv) traffic patterns (e.g., sensors producing large amount of push-up data towards the network versus servers providing push-down data or video streams); (v) activity patterns (e.g., always on versus long sleep mode devices).

Despite their diversity, resources are expected to collaborate at a certain degree to compose a unified infrastructure where network functions and end-user applications can be deployed on. This leads to the scenario where 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 whole 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 (a) to run and (b) being migrated everywhere in the network without incurring in 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 like cyber-physical applications (e.g., robotics) or event-driven software paradigm (e.g., cloud-based telemetry). According to this pattern, an application/network function (i.e., the publisher) decides when to push data into the system which is then delivered to other applications/network functions (i.e., the subscribers).
- **Data are computed on demand**: In this case, fresh data can be requested to applications/network functions whenever it is deemed necessary.
- **Data are stored**: Some data that has been published in the system requires to be stored temporarily or permanently to be processed later.
- **Data have different representations**: The format data are stored may be different from the format they were originally produced. In other terms, the format used when producing data may be different from the format used when consuming them.

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

To better frame the above in a NI-native architectural model such as that proposed by DAEMON, we can start from the observation that data produced in the network can be used for different purposes: examples include on-the-wire transmission optimization, local traffic and capacity forecast, anomaly detection, proactive replicas for redundancy, 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 produce some action as result. These models are seldom interested in how to retrieve and maintain the 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 having to deal 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 any NI model. To that end, the envisaged requirements are:

- **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 and fog 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 and fog infrastructure.
- **Consistency** 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 [6].

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 run at different locations in the network to accomplish an overarching goal. In case of a general NI algorithm, we can identify two main modes of operation: NI control and NI training. The former relates to the capability of running a trained model into the network, so as to control it at different timescales according to its intended purpose (e.g., resource scheduling, link optimization, capacity forecasting, etc.). The latter relates to the capability of consuming data and produces a model as output to be later deployed 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 B5G, 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 the end-to-end performance without having a detailed view on the status from individual domains.
2) The domain controllers that take local decisions for the domain they control (e.g., setting network schedulers, allocating compute resource for processes, routing traffic), both taking the interaction with adaptive applications into account.

Closing the control loops may lead to instability, if the control loops are not designed adequately. Currently, decisions are taken prudently (and often manually) to avoid instability issues. Moreover, the control loops should be stable under changing conditions (e.g., the introduction of a new software release). Our goal is not necessary to improve the performance of algorithms, but rather the level of automation. A reasonably good enough algorithm that is likely to be more stable in changing circumstances is preferred over an algorithm that is over-engineered to cope with only 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 periods of elevated traffic load that users of the network service produce (which fluctuations might impact the orchestrator decision). The timescale at which the domain controllers need to make decisions is much shorter (seconds to sub-seconds). As explained above, the orchestrator will only have a limited view on the details in the domain. Similarly, all desired data upon which the controller needs to make decisions will possibly not be available timely. Therefore, also the domain controllers will 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 still can make beneficial use of it to make sensible decisions without requiring huge information flows, and to isolate the right data in the domain that allows the domain controller to make timely decisions.

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 their 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 controller and make the scaling and resource provisioning decision 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-hierarchy one.

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

Last but not least, the aforementioned high-level per-network service intent itself shall 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 view 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, statistics, active end-users), whereas the stateful take the network service status into consideration and make the comparison correspondingly. To sum up, different intent comparison mechanisms shall be considered within 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 short timescale (us-ms)**: this may refer 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

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

large amount of space required. However, storing data in a small cache could be used to optimize the local system according to the recent history. The system is mainly characterized by high-throughput data producers and consumers. Data producer, consumer and storage need to be colocalized for performance reasons. The task to be accomplished is very specialized leading to a very well-defined dataset and optimization procedures. The NI models target exclusively on-line optimization. A data pipeline may be used to run multiple and independent NI models in parallel for redundant decisions.

- **Short timescale (ms-s)**: this case is like the previous one with the exception that data producer, consumer and storage may be distributed across neighboring nodes. The dataset and type of optimization procedures are still very specialized. However, at this timescale they can take the flavor of a local collaborative optimization with interaction between near components and systems. 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 could be run in a limited portion of the overall system. 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 a reactive/proactive optimization of the mobile network. In this case, a balanced mix of data producers, consumers, and storages are considered to both quickly reacting to events in the network and operating in response to the recent history of the system. Multiple NI models could be started to be chained together in a decentralized data pipeline where data are processed locally but decisions are taken at localized system level. The NI models target both on-line and (limited) off-line optimization. A data pipeline may integrate fresh and historic data for optimization.
- **Long timescale (min-h)**: optimizations at complete system level start to be feasible. Datasets are heterogeneous and composed of aggregated data. In this case, the data storages are the main source of data whereas data producers serve the purpose of updating those storages with aggregated data. A decentralized data pipeline could serve the purpose of aggregating 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 long timescale (h-days)**: at this timescale only system level optimization is performed. Data storages are the only source of data. Dataset and type of optimization procedures are extremely heterogeneous. Optimization is performed purely off-line. A data pipeline may include a human-in-the-loop.

Table I presents a summary of the different time scales characteristics. An additional important aspect related to control timescales is the capability of the control system to operate in a timely fashion, guaranteeing a response within a specified timing constraint. From a system-level point of view, 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*.

TABLE I: Summary of NI control timescale characteristics.

NI Control Timescale	Time units	Optimization level	On-line/off-line
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

III. Network-driven specifications for NI

Network-driven specifications of NI design are especially linked to the time dimension: aspects such as data collection or data processing depend on the underlying infrastructure and its capability to complete the actions, or directly on the time needed to execute the NI algorithms, yielding to 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 level (e.g., core, transport, edge, far-edge). This could be monitoring data coming from the network functions themselves, from the infrastructure hosting them, or 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**: the frequency at which data is gathered. Higher sampling rate force the infrastructure to have more capacity to support his.
- **Dataset richness**: 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.
- **Data locality**: monitoring data can be sampled from specific location of the network or from the entire network. A similar concept can be applied to the same groups of functions: e.g., collect KPI throughput from both the radio and the core together or just from the core.

The **infrastructure decision enforcement** includes all the parameters that can be changed, reconfigured and orchestrated, typically in a programmatic way through Application Programming Interfaces (APIs). Analogously to the monitoring data, the enforcement may happen at any level in the network.

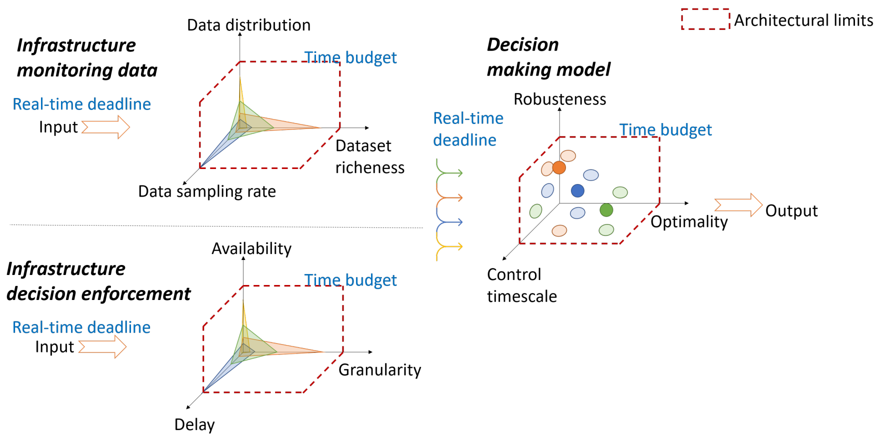

Fig. 2: Network Intelligence degrees of freedom, input-output relationship, and real-time constraints subject to infrastructure observability and controllability.

Again, this dimension can be split into three sub-dimensions, according to their trade-offs:

- **Delay**: the time it takes to generate the output of the algorithm and inject it through the API into the specific asset (Network Function or Infrastructure). This time is consumed by the budget available for the action.
- **Availability**: actions may be enforced at every single opportunity or may be resilient to some missed opportunity. For instance, scheduling patterns may be enforced at every Transmission Time Interval (TTI) or may be performed at coarser time granularity.
- **Granularity**: the granularity of the application of the given action. For instance, the maximum Modulation and Coding Scheme that could be applied at a single base station level 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 UE level but cannot enforce it again at UE level, if not on a subset of the gNBs. Or, if the algorithm requires a 1ms sampling granularity and 1ms-bounded action enforcement, then the gathered metrics and the configurations that can be applied to the network infrastructure could only be very fine if applied very locally to the same network function.

Once the timing constraint is in place, according to the specific network problem that has to be solved, then a class of NI is selected and implemented. However, there is a further degree of variability that has to be taken into account, which is intrinsic to the selected **model/algorithm for decision-making**. Once more, this can be articulated into three sub-dimensions:

- **Robustness**: this value indicates the resiliency of the NI to missed inputs. For instance, a suitably trained Neural Network 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 not applicable when inputs are missing due to, e.g., delays in the monitoring infrastructure.
- **Optimality**: the robustness and the precision that data driven solutions such as the one based on Neural Networks may achieve come 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. Hence, for some tasks, a trade-off between the optimality of the decision and the time needed to take it (e.g., with a simpler model or directly with a traditional optimization algorithm) shall be considered.
- **Control Timescale**: This dimension is similar to the infrastructure availability, which poses an upper bound to this value. NI algorithm can then decide to exploit all configuration opportunities or not.

Similar to the previous scenarios, these 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 optimization, that stay at the different ends of the spectrum: resilient and slow the former, precise but potentially more fragile and certainly faster the latter.

IV. Design principles of NI orchestration

Based on the requirements and specifications drafted in Sections II and III, we finally propose some initial guidelines for the design of an end-to-end closed control-loop of NI instances via a 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 Orchestrator, 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

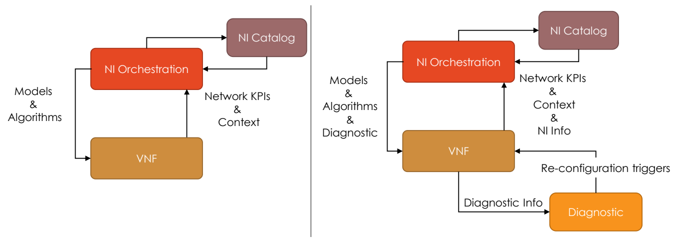

Fig. 3: Two different strategies for control loops between the NI instances and the NI Orchestration layer.

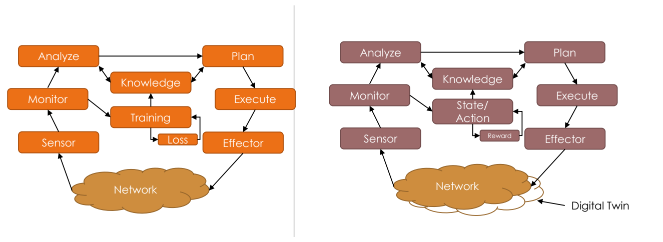

Fig. 4: Extended MAPE-K abstractions for NI algorithms based on supervised (left) and reinforcement (right) learning.

of a NI-agnostic interface between the NI instances and the NI Orchestrator.

A. Looping NI instances and Orchestrator

NI models are trained with data gathered from the network, and they possibly keep learning online, 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, 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, envisions a loops between the NI Orchestration (i.e., the element that manages the NI in the network, deployed at different domains). The interaction has two directions:

- From NI Orchestration to the Network Function: the NI Orchestration picks the best NI model and algorithms to perform a specific tasks, such as the one described in this document
- From the Network Function to the NI Orchestration, reporting on the achieved KPI (which are usually optimized by the NI) and some information about the context (i.e., the network conditions observed at the moment). This information allows the NI Orchestration 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 case, that require a tighter interaction between the “drift” control and its actual implementation. Thus, in a second approach (depicted in the right hand side of Figure 3) the NI orchestrator is sending an additional diagnostic module that can perform such computations on the drift very close to the network functions, trying to maximize the performance of the system at any point in time.

B. Unified representation of NI algorithms

In order to fully integrate the 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 [9].

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** includes the specific NI algorithm that is implemented, for instance a Neural Network 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** include the specific configuration parameters we are 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.

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 or reinforcement learning. For the former, depicted in the left part of Figure 4, 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). This module is replaced with **State/Action** representation and the reward table for reinforcement learning NI algorithms, as illustrated in the right part of Figure 4. Additionally, the effector and the sensors can be also redirected to a **Digital twin** element, if needed by the specific NI instance.

In Table II, we provide the extended MAPE-K definition for the vrAIn [7] and ATARI [8] algorithms as examples of

TABLE II: The MAPE-K definition for the vrAIn [7] and ATARI [8] 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.

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 APs and select the one that maximizes the performance of the deployment (i.e., solving the ChannelBonding problem)..

Ultimately, using the proposed approach allows specifying different kinds of input data, different ways of training a neural network (supervised learning or reinforcement learning) and a re-configuration 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 Native NI Architecture.

V. Conclusions and Future work

In this paper, we elaborated on the DAEMON architectural model [4], which proposes introducing a NI Orchestration layer for the effective end-to-end coordination of NI instances deployed across the whole mobile network infrastructure, and we have 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 have 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 evolving the NI Orchestration layer and applying the extended MAPE-K model to other NI instances.

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] 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

[6] 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

[7] 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.

[8] 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

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