# NetLabeller: Architecture with Data Extraction and Labelling Framework for Beyond 5G Networks

Jimena Andrade-Hoz, Jose M. Alcaraz-Calero, and Qi Wang

***Abstract*—The next generation of network capabilities coupled with artificial intelligence (AI) can provide innovative solutions for network control and self-optimisation. Network control demands a detailed knowledge of the network components to enforce the correct control rules. To this end, an immense number of metrics related to devices, flows, network rules, etc. can be used to describe the state of the network and to gain insights about which rule to enforce depending on the context. However, selection of the most relevant metrics often proves challenging and there is no readily available tool that can facilitate the dataset extraction and labelling for AI model training. This research work therefore first develops an analysis of the most relevant metrics in terms of network control to create a training dataset for future AI development purposes. It then presents a new architecture to allow the extraction of these metrics from a 5G network with a novel dataset visualisation and labelling tool to help perform the exploratory analysis and the labelling process of the resultant dataset. It is expected that the proposed architecture and its associated tools would significantly speed up the training process, which is crucial for the data-driven approach in developing AI-based network control capabilities.**

***Index Terms*—Data wrangling, labelling tool, networking dataset, self-optimisation, 5G.**

## I. INTRODUCTION

THE next generation of networks (NGN) is demanding a very sophisticated network control to advance 5G and beyond network architectures. Modern network control techniques boost concepts such as multi-tenant network isolation, network slicing, mobile network management, fine-grain quality of service control and firewalling capabilities. Artificial intelligence (AI) is having a crucial role to play in the development of such network control, allowing these network capabilities to develop in an optimal way. In this context, AI algorithms empower networks with the ability to autonomously detect, heal, and protect themselves, ensuring seamless and uninterrupted service delivery. This is why sixth generation (6G) cellular networks are expected to utilise a wide range of AI services from the core network to end users focused on the continuous optimisation of 6G networks [1]. This is achieved by following an AI-driven approach where intelligence will be an intrinsic feature of the 6G architecture [2].

Manuscript received November 9, 2022; revised November 9, 2023; approved for publication by Rehmani, Mubashir Husain, Division 3 Editor, November 11, 2023.

This work is funded partly by the European Commission under two projects: 6G BRAINS: Bringing Reinforcement learning Into Radio Light Network for Massive Connections [grant number H2020-ICT-2020-2/101017226] and ARCADIAN-IoT: Autonomous Trust, Security and Privacy Management Framework for IoT [grant number H2020-SU-DS-2020/101020259].

The authors are with School of Computing, Engineering & Physical Sciences University of the West of Scotland Paisley, United Kingdom, emails: {Jimena.andrade-hoz, Jose.alcaraz-calero, Qi.wang}@uws.ac.uk.

J. Andrade-Hoz is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000063

Self-optimisation capabilities in 5G networks require advanced network control. Such control involves the dynamic generation of network policies and the subsequent execution of those policies, requiring less human intervention and effort [3], [4]. The anatomy of such policy (or control rule, indistinctly) consists mainly of two differentiating parts. Firstly, the matching part is used to determine which packets/network flows the rule will apply to. Secondly, the targets are used to determine what will be done with those packets/flows [5]. Examples of control rules are performing a drop to a malicious traffic, prioritising specific traffic flows, setting concrete bandwidth limitation or guaranteeing a particular bandwidth to specified traffic. The dynamic generation of such rules imposes a level of complexity for the understanding of these rules that is very difficult to process in acceptable response times even for the most expert network administrators. This problem is exacerbated when it is foreseen the ultra-fast next-generation network such as 100GbE networks where the latency budget is very reduced, in terms of 1 nanosecond.

As a result, our network control policies are far from being optimised for several reasons. First, the vast majority of network control systems rely on the usage of only one particular datapath technology to perform the enforcement of such control, for example, OpenFlow [6] or XDP (eXpress data path) [7]. By doing so, these solutions are simply ignoring the benefits or drawbacks that can be brought by using a wider set of technologies available for the same purpose. Examples of these are Linux traffic control (TC) [8], Netronome BPF offloading, Mellanox offloading and Linux iptables [5], among others. Second, when network control rules are enforced into the system, they are already applied and considered the best decision taken. This fact clearly ignores the possibility of optimising such already enforced rules by i) making use of a more efficient datapath technology; ii) rewriting the rule in a more efficient way; or iii) reallocating the rule in a more efficient place. Third, the selection of the datapath technology used to enforce the rule, if any, simply ignores the current state of the system. Leveraging insights into the network's current state, one can make informed decisions to effectively utilise existing resources while simultaneously maximising the quality of user experience. And fourth, it is crucial to note that the decision-making process regarding the selection of datapath technology often neglects the significant considerations of bandwidth, packet losses, and latency associated with the chosen technology. Consequently, situations arise where the desired quality of services cannot be fulfilled due to decisions regarding technology selection.

The use of AI can significantly address this current deficiency in optimising network policies. AI can make the continuous optimisation of control rules feasible when it is combined with a high level of automation [9]. In this way, the AI can generate decisions considering all available possibilities. Not only considering all possibilities related to the type of technology to use, but also the state of the network at any given moment. This AI-native approach enables the network to be agile, intelligent and able to learn and self-adapt to changing network conditions [10]. In addition, these decisions will be made much faster than if compared to any network expert. This speed is essential, especially when these network policies are intended to deal with network attacks, one of the most important concerns of 5G operators nowadays [11].

To achieve so, it is essential to first learn which datapath technology is the most appropriate for each specific use case. This decision will depend on multiple factors, in particular, the state of the network at any given time, the type of rule to be enforced and the number of rules currently enforced in every datapath technology. For example, in a use case were a drop rule is desired to be optimised, we will prioritise speed of execution, choosing datapath technologies located at the lowest point in the software stack, such as XDP [12]. Another example in the same scenario, taking into account other criteria such as having a lot of rules enforced in XDP, we could optimise the insertion of a new rule using a faster one such as Open vSwitch (OVS) [13]. Considering all these variables and the range of applicable use cases in a beyond 5G network, it is imperative to have a novel framework with the necessary tools to monitor it. This will allow to extract as much information as possible regarding network status, control and management [14]. Developing a framework with these characteristics will allow to access to network metrics, metadata, network topology, and rule-specific information, thereby enabling the generation of comprehensive datasets. These datasets will enhance the knowledge of what is the state of the network and how is it performing. Subsequently, such datasets can be utilised to extract multitude of insights. Also, to identify potential patterns that can facilitate the labelling process in terms of the datapath technology selection. Finally, the generated datasets would be used for training AI models for the creation of more optimised rules. To the best of our knowledge, we have not encountered any existing framework that adequately addresses the extraction, storage, and labelling of the current state of a network infrastructure. This knowledge gap can be attributed, in part, to the complexity involved in real-time extraction of underlying metrics, coupled with the need for a comprehensible tool tailored specifically for network engineers to interpret such information. Moreover, it is crucial to highlight the challenges associated with extracting metrics and topology in a heterogeneous network environment like that of a 5G network.

The previous challenges have motivated this research work. The main contribution of this manuscript is multi-field. First, the creation of a data extraction framework able to compile real-time network-related datasets from the most novel networks available today including cloud infrastructures, mobile edge computing infrastructures and beyond 5G networks, among others. Second, to perform a deep analysis of the features and metrics that are involved in the selection process of a particular datapath technology of network control. The main intention is to allow the identification of all possible relevant metrics gathered from network devices, network flows, network rules and other contextual elements required to take an informed decision in the network. Third, this analysis is complemented in a tailored labelling tool specially designed to perform the visualisation and labelling of network-related datasets. This exploratory data analysis (EDA) process is used to visually represent the knowledge embedded deep in the extracted data from the network. The proposed technique helps to generate inferences from the resulted dataset. The following list enumerates the contributions provided with respect to the state of the art:

1) To provide a novel architecture to allow the extraction, analysis and labelling of data from a live beyond 5G network for future training of an AI model to perform optimised decisions.
2) To perform the analysis of different network features among the immense number of available metrics in the network related to network flows, network devices, network queues, network rules, etcetera.
3) To provide a novel mechanism to allow the visualisation of network-related datasets that will help to perform the labelling process to enable the training of AI models based on real testbeds.

To describe the main contribution, this paper has been layout as follows: Section II provides a state of art in available frameworks for visualising and labelling network-related datasets. Section III introduces the architecture proposed to fulfil our contribution along with an overview of data flow in a 5G/Pre-6G network infrastructure. This section provides the reader with an easy way to understand our work within a NGN. Section IV provides the analysis of the most relevant network features that will compose the 5G networking dataset. Section V presents the design of our contribution, emphasising the workflow and the necessary elements for the correct functioning of the framework. Section VI provides the implementation details in terms of hardware and software used for the deployment of the framework and the testbed used for executing the experiments. Section VII presents some analytical experiment results of the framework proposed and its performance. Finally, conclusions are drawn with future research activities outlined in Section VIII.

## II. Related Work

Supervised machine learning algorithms use pre-labelled datasets to train models. This fact leads to the need for tools that make the labelling process as fast, easy, accessible and effective as possible for data engineers. Table I compares the key features addressed in this research work that are essential

TABLE I
COMPARISON OF DIFFERENT LABELLING TOOLS (NP: NOT PROVIDED).

| | | Data visualization | | | | Data labelling | | | Dataset type | | | | | | Input | | Output | Other | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| I | [15] | ✓ | ✓ | ✓ | × | × | - | - | × | ✓ | ✓ | × | ✓ | × | × | n.p | - | ✓ | n.p |
| | [16] | ✓ | ✓ | ✓ | × | × | - | - | × | ✓ | ✓ | ✓ | ✓ | × | × | - | - | ✓ | n.p |
| | [17] | ✓ | ✓ | ✓ | × | × | - | - | × | ✓ | ✓ | ✓ | ✓ | × | × | CSV | - | ✓ | BSD-3-Clause |
| II | [18] | × | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | ✓ | ✓ | × | Image, video and text | PascalVoc XML, CoreNLP | ✓ | n.p |
| | [19] | ✓ | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | ✓ | × | Audio, HTML, Images, Paragraphs, CSV, TSV, txt, json | Json, NumPy 2d arrays, PNG images, COCO, CoNLL2002, CSV, etc | ✓ | Apache 2.0 |
| | [20] | × | ✓ | ✓ | × | ✓ | ✓ | ✓ | × | × | × | ✓ | × | ✓ | × | Text, tabular, image, and log data | CSV | ✓ | Apache 2.0 |
| | [21] | × | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ | × | × | × | ✓ | ✓ | × | Image, video | Image formats | ✓ | Open source, enterprise |
| III | [22] | ✓ | ✓ | ✓ | ✓ | × | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | n.p | CSV, JSON, MySQL, MongoDB, Elasticsearch, AWS CloudWatch, Azure monitor, GC monitoring, etc. | - | ✓ | AGPLv3 |
| | [23] | ✓ | ✓ | ✓ | ✓ | × | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | n.p | Elasticsearch data | - | ✓ | Elastic license 2.0 |
| | [24] | ✓ | ✓ | ✓ | ✓ | × | - | - | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | Google apps, CSV, Amazon Redshift, MySQL, PostgreSQL, etc. | - | × | n.p |
| IV | [25] | ✓ | ✓ | ✓ | × | ✓ | × | n.p | × | ✓ | ✓ | ✓ | ✓ | × | × | n.p | n.p | ✓ | MIT |
| | Our | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × | ✓ | ✓ | ✓ | ✓ | × | ✓ | CSV, JSON, MySQL | CSV | ✓ | Apache 2.0 |

TABLE II
DESCRIPTION OF THE CHARACTERISTICS ANALYSED IN TABLE I.

| Clasification capabilities | | | Description |
|---|---|---|---|
| Data visualization | 1 | Graphs | The tool has the functionality to add graphs. |
| | 2 | Data exploration visualisations | The tool allows the user to explore and interact with the data being displayed. |
| | 3 | Multi-feature dashboard | The dashboard allows displaying multiple features. |
| | 4 | Dashboard personalisation | The tool allows the user to personalize the dashboard. |
| Data labelling | 5 | Dataset labelling | The tool allows the user to label the dataset displayed. |
| | 6 | Cooperative labelling | The tool has functionalities that enable collaborative labelling. |
| | 7 | Automatic label navigation | During the labelling process, the tool automatically moves on to the next instance to be labelled. |
| Dataset type support | 8 | Multi-modal data | The dataset contains features coming from more than one data representation mode. |
| | 9 | Multi-feature data | The dataset contains more than one feature. |
| | 10 | Time-series data | The dataset contains features coming from a series of data points indexed in time order. |
| | 11 | Discrete feature data | The dataset has features that come from a limited number of values. |
| | 12 | Continuous feature data | The dataset has features that come from an unlimited number of values. |
| | 13 | Multi-media | The dataset has features that come from image, audio, animation, graphics, etc. |
| Input | 14 | Streaming support | The dataset is being consumed from a streaming data source |
| | 15 | Format | Describes the different formats the dataset can have. |
| Output | 16 | Format | Describes the different formats the exported dataset can have. |
| Other | 17 | Open source | Describes whether the software is designed to be publicly accessible. |
| | 18 | License | Expecifies the type of legally binding guidelines for the use and distribution of the software. |

to be present in a labelling framework in order to fulfil the innovations characteristics mentioned in the previous section. Such comparison is done with respect to the state of the art and the analysis presented has led to the identification of some gaps described below. The features have been classified into six categories: Data visualisation, related to the way that data is visualised; data labelling, related to the fact whether the tool has the functionality to label datasets or not; dataset type, describing the different data types that are possible to represent and label using the framework; input: related to the data source given as input to the framework; output: describes the labelled dataset formats; and finally, open source and license. Each category subtypes are numbered from 1 to 18 due to space limitations and their respective description together with an explanation of the feature is shown in Table II. Table I also shows a logical grouping of some of the analysed research works using roman numbers for such grouping purpose. The information that has not been provided is represented in the table as n.p (not provided).

The first group in Table I represents some recent research focused on the visualisation and exploration of different-purpose datasets. For example, Stopar *et al.* [15] presents an interactive web-based visualisation tool called StreamStory that helps to interpret and analyse data coming from time-varying systems such as weather, GPS and wind data. The purpose is to provide a high-level conceptual summary of such data, allowing users to interactively identify and interpret large multivariate time series datasets. Another example is the provided by Philip *et al.* [16], whose work presents an exploratory and visual data analysis for diabetes disease datasets. The analytic is implemented in the R programming language and represents the first steps in a framework development, but it has not been implemented at the end. Finally, an EDA process of COVID-19 data was carried out by Dsouza and Velan S. [17] using the Python libraries MatplotLib and Seaborn. These three research works reveal two key points: i) There is a tendency to perform EDA on datasets as a step before labelling data and ii) the lack of tools that allow visual analysis and labelling of the data being observed in the same software.

Therefore, the next group analysed (see group II in Table I) corresponds to specific labelling data tools. Four different labelling tools have been examined: Colabeler [18], Label Studio [19], DAML [20] and Diffgram [21]. As is shown in Table I, some of them allow to include graphs, and user interactive visualisations. Furthermore, multi-feature dashboards are common in all of them. Remarkably, all of them are very sophisticated in terms of data labelling characteristics, allowing them to be used by more than one person at once or to move to the next item to be labelled automatically. Finally, recall that all these tools are open source. The results show that there is high popularity in labelling tools whose purpose is to label multi-media datasets, such as images, videos and text, but there is a clear lack of labelling tools whose purpose is other different from these three ones.

For the reason mentioned above, some visualisation tools (see group III in Table I) have been examined in which it is possible to create personalised dashboards. Visualisation frameworks such as Grafana, examined by Chakraborty and Kundan [22], Kibana, studied by Sharma [23] or Google Looker Studio, Snipes [24], are appropriate in terms of visualising time-series, discrete and continuous data in complex dashboards. Furthermore, they allow a wide range of input data types. These frameworks would be suitable for our purpose if it were not for the fact that they do not allow the labelling of the data being displayed. Finally, Chegini *et al.* [25] present a system called mVis as a visual analytical approach that allows interactively labelling datasets. This system is the closest to what it is being looked for in this research because it facilitates interactive visual interfaces for data exploration, allowing the meaningful selection and labelling of records based on insights gained by the user. The tool has many graphs to represent the data, but on the downside, it is not possible to customise a dashboard or add data that is not representable in a form of a graph.

Finally, a further search not only for labelling tools, but also for frameworks that included data extraction and network labelling has been carried out. The explicit methodology for searching such related work has been the following. First, the search was conducted in major scientific research sources in the field of Computer Science, namely, IEEE Xplore, Elsevier via Scopus and SpringerLink. In addition, we performed the same search on GitHub, for extending the search to code-based results. Second, we performed the same advanced search with the following terms: “Network dataset” AND “labelling tool” AND “labeller” AND “5G”. We restricted the results to articles and conferences in the period of the last 10 years, i.e., from 2014 to 2023. The results obtained were as follows. Elsevier obtained 95 results, after reading their title, any reader can easily see that none of them are related to our work and that the search engine is not as accurate as other engines. The same was the case for the 2 results obtained on SpringerLink and the other 2 on GitHub. All of them were out of the context of our research work. Two results relevant to our subject were found on IEEE Xplore. Lee *et al.* presented in [26] a network collector system, data analyser and a 5G-based labelled dataset. Similarly, Lee *et al.* proposed a system that performs data labelling, filtering, preprocessing, and learning for 5G network flow and security event data [27]. Despite being closely related strategies, the authors do not present an in-depth analysis of these frameworks’ performance.

As the reader can see, from the state of the art no research work delivering a proper framework that facilitates network data extraction, visualising and labelling has been found. This has been the main motivation of this research work together with the impact associated with our proposal.

## III. Overview of the Proposed Architecture for Data Extraction and Labelling

This section presents the proposed architecture over a Beyond 5G network infrastructure. Fig. 1 has been divided into two parts. At the bottom of the figure, the reader can see the managed infrastructure where a 5G network architecture in a multi-tenant environment is depicted. At the top of the picture is the management infrastructure. Each of them are described in detail in the following subsections.

### *A. Managed Infrastructure*

The managed infrastructure presents a 5G network where physical resources are shared through virtualisation by multiple tenants (e.g., network operators) in a multi-tenant environment. Virtual tenant networks are depicted with different colours representing the isolation of each tenant’s virtual infrastructure. Five different network segments are shown: Radio access network (RAN), edge network, transport network, core network, and inter-domain network. The RAN segment is part of the 5G network communication system infrastructure that facilitates the connection between UEs (user equipments) and the core network [28]. It corresponds to a distributed collection of antennas and radio units (RU), where the radio frequency signals are transmitted, received, amplified, and digitised. Each RU has its own distributed unit (DU) associated, which is connected to a centralised unit (CU). These two components form what is called the gNB (next-generation node B) and form the computation parts of the base station, sending the digitised radio signal into the network. They handle the control plane and the data plane through different interfaces. There is a single CU for each gNB, but one CU controls multiple DUs. As shown in Fig. 1, the gNB is located in the Edge

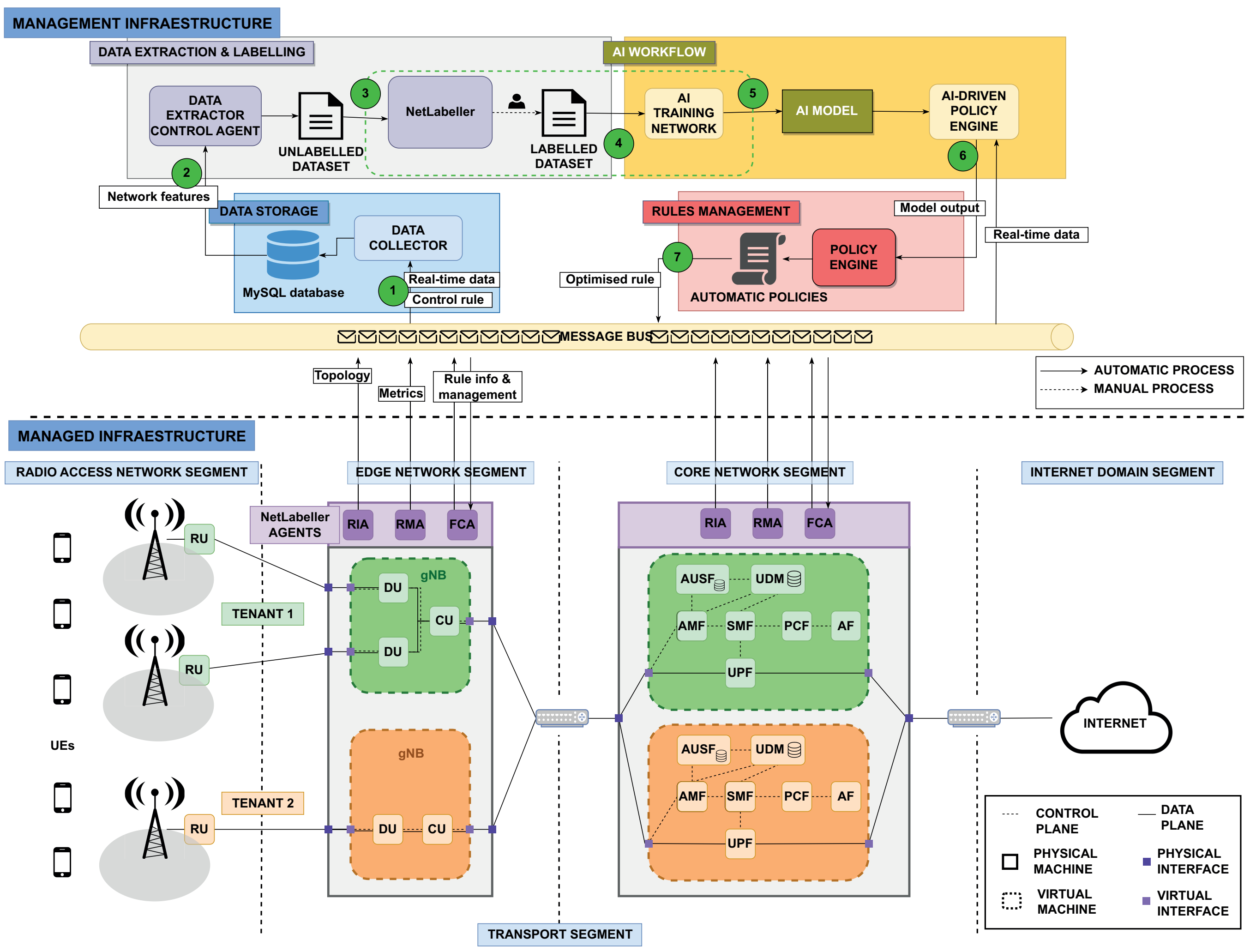


Fig. 1. Overview of the proposed data extraction and labelling architecture in a 5G multi-tenant infrastructure.

Network segment. The Edge constitutes geo-distributed servers with virtualisation capabilities that provide an IT service environment and cloud computing capabilities at the edge of the mobile network [29]. Deploying edge computing at the base station enhances computation and avoids bottlenecks and system failure [30].

Edge and core networks are connected through the Transport Segment. The core network segment is where access for the user to other networks is provided. The 5G core network functions are divided into seven components, split up by service. Six of them handle the control plane: Core access and mobility management function (AMF), session management function (SMF), policy control function (PCF), application function (AF), authentication server function (AUSF) and user data management (UDM). In summary, these components are responsible for UE authentication, UE management and IP addresses, session management for the data transfer, policy control and UE data storage. Finally, the user plane function (UPF) handles the data plane and provides Internet access or operator services [31].

Above the physical infrastructure, the NetLabeller agents are placed. These components are part of the management infrastructure but, as is shown in Fig. 1, they are deployed in the managed infrastructure as agents distributed across the infrastructure. For this reason, they will be explained subsequently.

## B. Management Infrastructure

Three different types of NetLabeller agents are deployed in each of the computers available in both Edge and Core (see purple boxes in the managed infrastructure of Fig. 1). They publish different network information using the RabbitMQ message broker. The first one is the resource inventory agent (RIA) presented by Sanchez-Navarro *et al.* [32]. It is in charge of publishing topological network information in real-time. It discovers both physical and virtual machines where the agent is deployed and their topological connections.

The second one is the resource monitoring agent (RMA). It allows the monitoring of metrics extracted from the physical machine where the agent is deployed and its virtual machines. The metrics to be monitored and published are configured in a configuration file and can be easily extended using such configuration file. This allows extensibility to our architecture, making it suitable for multiple use cases. Notice that such metrics are extracted from devices, device ports (physical and virtual network interfaces), processes, operating systems and datapath technologies (e.g., OpenFlow metrics, XDP metrics, Linux TC metrics, IPTABLES metrics, etc.), among others depending on the use case addressed. This agent makes use of the list of technologies available on each of the network ports previously discovered and published by the RIA.

The third one is the flow control agent (FCA) presented by Matencio Escolar *et al*. [33] as Slice Control Agent. It is in charge of exposing network traffic control functionalities. These functionalities can be divided into two. On the one hand, it provides a unified technology-independent control interface for enabling wired and wireless programmable datapath technologies, allowing the enforcement of rules for every packet that traverses the different datapath technologies in the network. On the other hand, it monitors and publishes the control metrics coming from the enforced rules. The topology, extracted by the RIA, the metrics, collected by the RMA and the rule metrics reported by the FCA are published into the RabbitMQ message bus as it is depicted in Fig. 1. RabbitMQ is an open-source message broker that manages asynchronous messaging. From this point, the published data follow a series of actions divided into 4 steps. It is important to highlight that this research paper exclusively focuses on the initial two steps, which are data storage, extraction and labelling. However, a comprehensive presentation of the infrastructure is provided to enhance the reader's comprehension of the proposed framework's utility. The steps are described below.

*1) Data storage:* The first step is the data storage where the network topology and metrics information are consumed by the data collector component (see 1 in Fig. 1). This component extracts the information published by the NetLabeller agents, adapts the data and stores it into a MySQL database. Note that the information associated with the control rule to be optimised is also stored. The architecture is generic enough to make use of any SQL and non-SQL database but it has been used MySQL for convenience. As a result, this step generates and updates a database with all the network information published in real-time. It is important to mention that both topology and metrics have associated metadata to each of the values in order to allow the identification of the producer of the data (see *monitoringResourceId* in Listing 1), the source of the data (*monitoredResourceId*), the time the data have been produced (*reportedTime*) and other relevant metadata information shown in Listing 1.

```
"MetricSample": [
    {
      "monitoringResourceId": "QG54TY8U",
      "monitoredResourceId": "YTR56HJ8",
      "metricName": "devicePortSpeed",
      "metricValue": "1000000",
      "startSamplingTime": 1659107373856,
      "resourceId": "4BA92944",
      "reportedTime": 1659106981511
    }
  ]
```

Listing 1: Example of the metadata stored in the database associated to a metric.

*2) Data extraction and labelling:* The next step involves both data extraction and data labelling where data is extracted, shaped, sorted and adapted to comma-separated values (CSV) format ready for AI training. To achieve so, firstly, the data extractor control agent (DECA) connects with the database and carries out a continuous loop of queries to extract the selected data periodically (see 2 in Fig. 1). Then, the data is sorted and written in a CSV file, which is represented as the unlabelled dataset in Fig. 1 (see 3). A new instance of the CSV file is generated each time the data is extracted. The dataset contains as many columns as features selected and as many rows as instances have been written. As we we will be facing a supervised machine learning problem, once the dataset is completed, it is necessary to label each instance of it. This process must be done manually by a network expert and ideally with the help of a tool that facilitates the visualisation of each of the values in the CSV file. The labelling tool NetLabeller is in charge of this task, allowing the visualisation of the values in a user interface that allows labelling each of the instances of the dataset. As a result of this phase, the dataset is labelled and ready to be used in supervised learning (see 4 in Fig. 1).

*3) AI workflow:* The next step is the AI workflow, which consists of three different stages. In the AI training network, an AI model will be trained with the training dataset. This step consists of adjusting the model over the training to fit the objectives that have been set. Once the desired accuracy has been achieved, the final AI model will be attained (see 5 in Fig. 1). Note that the labelling and AI training boxes are enclosed by dashed green lines, meaning that these steps will be done only during training. Then, the AI-driven policy engine will execute the AI model with the real-time data published by the NetLabeller Agents. As a result of this step, the output of the AI model will be obtained (see 6 in Fig. 1).

*4) Rules management:* Finally, the rules management step is composed by a policy engine that will generate an automatic policy. This automated policy leads to the generation of an optimised network rule (see 7 in Fig. 1). Then, such optimised rule is published in the RabbitMQ message broker. The FCA NetLabeller agent will consume this message and will enforce the rule where appropriate, closing the AI-driven self-optimisation loop.

## IV. Beyond 5G Networking Dataset for AI-driven Self-optimisation Purposes

### A. Running Example

In order to a better explanation of the usage of the proposed architecture for labelling, this section aims to indicate a concrete problem to be addressed by an AI model as a running example throughout the whole manuscript. It is worth indicating that the proposed architecture is generic and not tailored to any particular use case. Therefore, this proposed infrastructure holds potential for application in various other AI-driven use cases. However, it will improve the readability of the reader and will allow them to understand the complete methodology followed.

As introduced in the Section I, the most modern infrastructures today are making use of innovative datapath technologies for network control such as network accelerators (Mellanox, Netronome, Intel, etc.), kernel bypass technologies (DPDK, AF-XDP, etc.) on top of the traditional and well-established kernel technologies (traffic control, iptables, etc). This heterogeneity of technologies is no more a matter of choosing one of them to be used but now they all are available at the same time in the same network interface. It causes a lot of confusion as it is incredibly difficult for network administrators to determine which one is the best technology to implement a concrete network control rule. This is because it depends on a significantly wide number of parameters such as the current status of the system, the anatomy of the rule and its usage, and other related metrics and metadata. Thus, this is where an AI model can help to determine which one is the best technology to materialise a given network control rule in a given acceptable response time. For this particular example, the output of the model will be the optimal datapath technology for every enforced rule on the network.

## B. Analysis of Features

This section provides an analysis of different network metrics related to the edge and core of a beyond 5G network. The characteristics come from each network element (hardware and software) capable of reporting any information related to the control and data plane. This information includes the actual status about network flows, network devices, network interfaces, network queues and network rules.

As a datapath technology is desired to be selected depending on the state of the network, the vast majority of the metrics are associated with network interfaces. Such network interface will be henceforth referred to indistinctly as device port. Furthermore, each datatpath technology for network control available in each device port will be henceforth referred to as datapath point. Fig. 2 shows a physical network interface with some datapath point technologies represented as orange dots. It is worth noting that a high-level overview of the various stages of the input data path is provided. This allows the reader to better understand what a datapath point is and how data flow passes through them. The data flow goes through the network interface crossing each datapath point as indicated by the arrow. Every point represents stages where various operations can be performed on network packets. Note that the existing number of datapath technologies might vary when it comes to a virtual interface. The reader can see in Fig. 2 the firmware used to refer to hardware-offloading rules. The driver is to refer to those rules enforced before the packet arrives at the kernel network stack. Then, the traditional kernel hooks within the Linux kernel that allows packet filtering, network address translation (NAT), and other network-related operations. Examples are the libpcap interface to perform network monitoring, the Linux traffic control (TC) used to deal with quality of service rules, the OVS control and the iptables control.

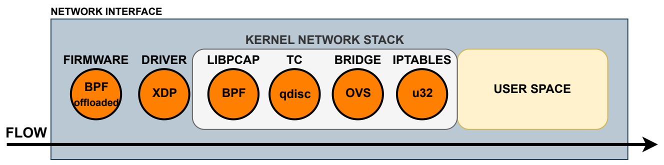


Fig. 2. Identification of the data flow through the different datapath points of a network interface (input path).

The feature analysis is done in a top-down approach, starting with device features, then device port features, datapath point features, queues and finally, features related to network flows. Fig. 3 shows a device, i.e., a physical or a virtual machine in purple, with a device port coloured in grey. The device port depicted in this example consists of two different datapath points coloured in yellow as a mere simplification to improve readability. Each datapath point may or may not have rules to be enforced for every packet that traverses it. These rules are represented in the figure in red. Arrows represent the direction of data flow: ingress and egress. Fig. 3 also presents 34 features located in both the bottom and the top part of the figure. They are coming from these components and are coloured with the same colour of the entity they belong to. This allows the reader to understand if they are metrics related to device, device port, datapath points, rules , queues or flows. They have been divided into control plane features and data plane features, which will be explained in detail in the following subsections.

## C. Data Plane Feature Metrics

Data plane metrics are depicted in the lower part of Fig. 3 (number 16 to 34). Within network metrics, we can distinguish between received (RX) and transmitted (TX) network metrics. These metrics can be bytes or packets. In the diagram, different types of queues are schematically represented according to the datapath point concerned. For instance, datapath point 1 has a single transmit and receive queue. For simplicity, the receive and transmit queues are represented as a single queue. On the other hand, as shown in the figure, datapath point 2 has two queues for reception and 2 for transmission.

Starting with flow metrics, the type of encapsulation and the sense is included. By analysing the encapsulation type and flow direction, we can determine the behaviour of different traffic types and understand how they impact network performance. Also, the size of each packet of the flow and the total packets is gathered. With the use of these two metrics we can calculate the overall data volume transmitted. This information helps in analysing bandwidth utilisation and identifying potential bottlenecks or congestion points. Focusing now on the metrics related to datapath points, both packets and bytes transmitted and received per second have been selected. Packets dropped describe dropped packets of data not reaching their destination during data transmission. This group of metrics gives us insights into whether there is network congestion. The packets dropped per second indicate

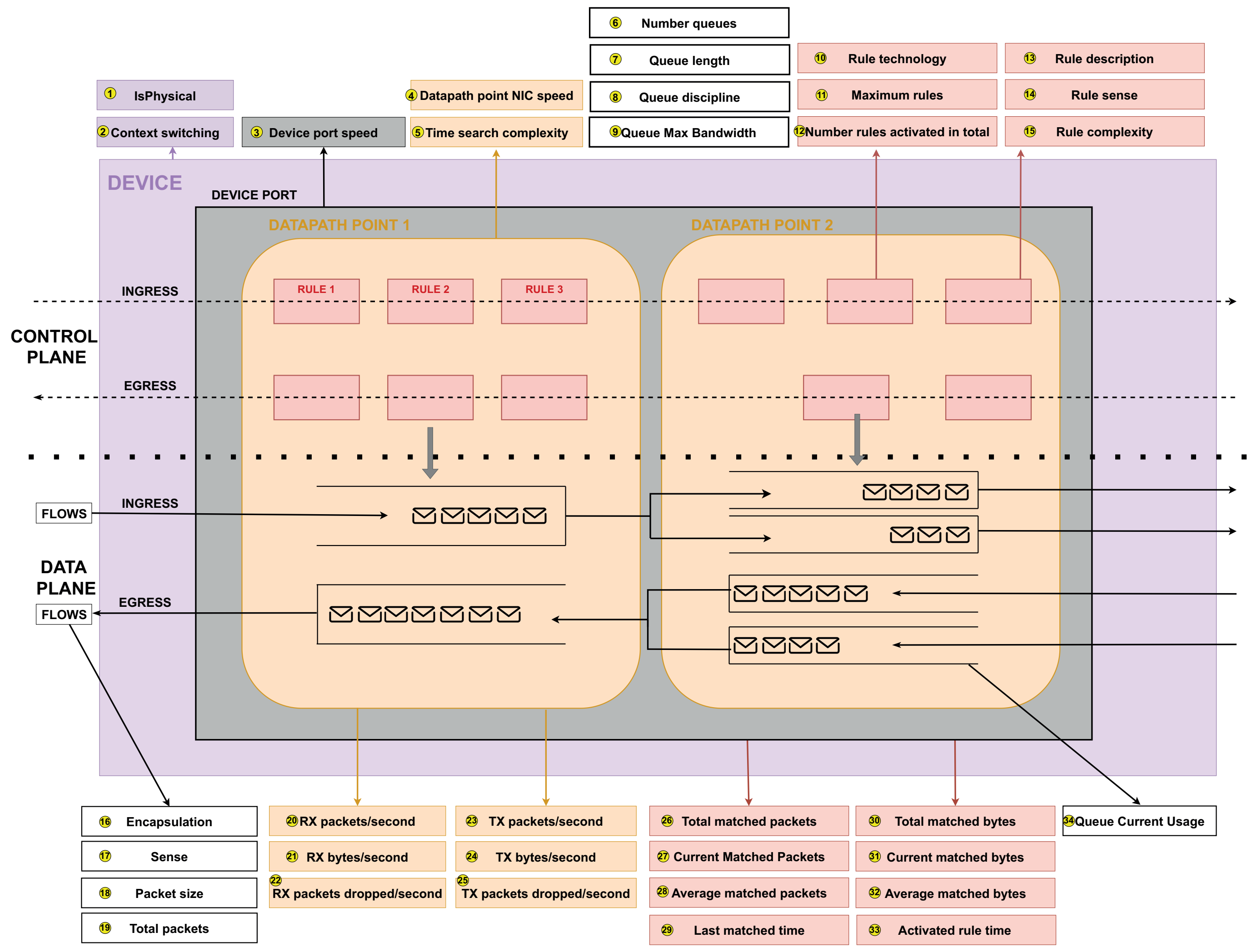


Fig. 3. Identification of 5G network features in different network resource levels: device, device port, datapath point, flows and queues.

if the network traffic is hitting its maximum limit at the moment, while the total packets dropped specifies the idea of how often the network reaches its limit and needs to drop packets. Regarding rule metrics, it is essential to know how many packets and bytes are matching each rule in total (total matched packets and bytes), at the moment (current matched packets and bytes) and on average. These metrics can be compared with the datapath point metrics to determine what percentage of packets are being matched compared to all packets passing through the network. This gives us an idea of the rule efficiency. The last matched time indicates when the last time a packet was matched. Finally, the activated rule time specifies how many seconds has a rule been activated. We conclude the data plane metrics with the queue current usage, which describes how congested each of the queues of a datapath point is. This metric, when compared to the maximum values described in the control plane, can estimate whether the network is in a nearly congested state that will lead to packet loss.

## *D. Control Plane Feature Metrics*

Control plane feature metrics are depicted in the upper part of Fig. 3 (number 1 to 15). Device metrics are represented in purple. The IsPhysical represents whether the device is physical or logical/virtual. This feature has some relevance for technology selection, since it has already been mentioned above that certain datapath points are not found in virtual network interfaces. Device metrics also include context switching related to the number of times the CPU is performing context switching between processes. Device port metrics are represented in grey. The device port speed indicates the port's network interface card (NIC). Knowing the speed limit of the NIC can be helpful to diagnose performance issues related to the bandwidth and the performance expected. Datapath point metrics are depicted in yellow. The Datapath Point NIC Speed is included as it may differ from the device port speed. Indeed, this is a typical bottleneck that happens in today's networking stack where we do have a 10 Gbps card but the Linux kernel is only able to provide 4–5 Gbps due to limitations in any of the technology available. The time search complexity represents the difficulty in terms of the time it takes for a datapath point to locate a rule. Some datapath points have lists of rules, for example, TC or OVS, some others have hashmaps or ternary content address memory (TCAM) structures to deal with the time search complexity.

In addition, several data path point technologies provide support for network queues. Therefore, queue metrics are also included in the diagram, coloured in white. They indicate how many queues the datapath point has (number queues), the length of each queue (queue length), the maximum bandwidth of the queue and the queue discipline, which describes the scheduling algorithm determining the formation of a queue or queues [34]. The information related to the queues will give us an idea of the maximum capacities of the queues and their respective limits.

We conclude with the control features introducing the ones related to the rules. The rule technology specifies the datapath point technology, which could be any of those shown in Fig. 2 or another one. For our concrete AI model, this will be in fact the label we want to infer. The rule description includes extra information about the rule, such as the action type and the action name. To avoid reaching the maximum number of enforced rules, it is necessary to know the maximum number of rules that can be inserted in each datapath point. This value, together with the number of rules activated in total, will give an idea of how much percentage of rule resources are used/available. The rule sense indicates whether the packets to be affected are ingress or egress packets. Finally, the rule complexity represents a value that measures the complexity of the rule to be enforced (the longer, the most complex).

The reader can understand the effort underneath carried out in order to design, discover, understand, prototype and refine the system able to extract all these sources of heterogeneous features in order to allow to have enough data to be used for training purposes in AI-models. This is probably the first manuscript we have found in the literature with this deep reveal of the network metrics involved in the decision taking process of network control technology selection. It is worth remarking that all the metrics collected are related to L2–L7 of the OSI stack and there is not any attempt to gather L1 metrics, mainly relevant to wireless interfaces. This is because we are currently focused on Edge and Core network segments of the 5G infrastructure.

## V. Design of the Data Extraction and Labelling Components

As shown in Fig. 1, the management architecture is divided into four modules. This section will introduce in more detail the data extraction and Labelling module where the features described in the previous section are extracted, shaped and written in a dataset for a resulting EDA analysis and labelling process. This process has been analysed mathematically in subsection V-A. Then, subsections V-B and V-C describe the design of the two most relevant components, these are the data extractor control agent (DECA) and NetLabeller. Both components provide, as a result, a CSV file. The DECA produces a CSV file with all the metrics and metadata extracted in the network and the NetLabeller produces a CSV file with labels added to each instance. The labelled CSV file represents the networking dataset that will be used for AI purposes. Finally, in subsection V-D, a sequence diagram is presented to help the reader understand where all the data extracted by DECA comes from.

### *A. Mathematical Analysis of Proposed Architecture*

This section provides a pseudo-code-based algorithm for the proposed data extraction and labelling architecture. The primary focus lies in the dataset generation part. We first start with the definition of the origin of the different feature metrics as well as the resource they are associated with.

We define $\mathbb{M} = \{\mathbb{D}, \mathbb{P}, \mathbb{T}, \mathbb{Q}, \mathbb{F}, \mathbb{R}\}$ as the set of metrics associated to the device port $p$ where a rule $R$ is enforced. The specification of all these metrics is detailed below.

$\mathbb{D}$ corresponds to the $b$ metrics associated with the device $d$ on which the device port $p$ is located. It is given by the following equation:

$$\mathbb{D} = \{D_1^p, \cdots, D_b^p\} \tag{1}$$

$\mathbb{P}$ corresponds to the $c$ metrics associated with the device port $p$, given by the following equation:

$$\mathbb{P} = \{P_1^p, \cdots, P_c^p\} \tag{2}$$

$\mathbb{T}$ corresponds to the $f$ datapath technology metrics associated with the device port $p$ from the $x$ different datapath technologies available in $p$. It is given by the following equation:

$$\mathbb{T} = \{T_1^{1p}, \cdots, T_f^{1p}, \cdots, T_1^{xp}, \cdots, T_f^{xp}\} \tag{3}$$

$\mathbb{Q}$ corresponds to the $g$ queue metrics for each of the $q$ queues in the datapath technology $x$ from the device port $p$. It is given by the following equation:

$$\mathbb{Q} = \{Q_1^{11p}, \cdots, Q_g^{q1p}, Q_1^{12p}, \cdots, Q_g^{q2p}, \cdots, Q_1^{1xp}, \cdots, Q_g^{qxp}\} \tag{4}$$

$\mathbb{F}$ corresponds to the $h$ flow metrics associated with the $i$ flows affected by the rule $R$ in the device port $p$. It is given by the following equation:

$$\mathbb{F} = \{F_1^{1pR}, \cdots, F_h^{1pR}, \cdots, F_1^{ipR}, \cdots, F_h^{ipR}\} \tag{5}$$

$\mathbb{R}$ corresponds to the $j$ rule metrics associated with the rule $R$ and enforced in the device port $p$. It is given by the following equation:

$$\mathbb{R} = \{R_1^{pR}, \cdots, R_j^{pR}\} \tag{6}$$

The sets $\mathbb{D}$, $\mathbb{P}$, $\mathbb{T}$ and $\mathbb{Q}$ are provided by the RMA agent. The sets $\mathbb{F}$ and $\mathbb{R}$ are provided by the FCA agent.

Let $I_t^{pR}$ be the instance corresponding to the extraction at time $t$ of $\mathbb{M}$, given by the following equation:

$$\begin{aligned} I_t^{pR} = \{&D_1^p, \cdots, D_b^p, P_1^p, \cdots, P_c^p, T_1^{1p}, \cdots, T_f^{1p}, \cdots, T_1^{xp}, \cdots, \\ &T_f^{xp}, Q_1^{11p}, \cdots, Q_g^{q1p}, Q_1^{12p}, \cdots, Q_g^{q2p}, \cdots, Q_1^{1xp}, \cdots, \\ &Q_g^{qxp}, F_1^{1pR}, \cdots, F_h^{1pR}, \cdots, F_1^{ipR}, \cdots, F_h^{ipR}, R_1^{pR}, \\ &\cdots, R_j^{pR}\} \end{aligned} \tag{7}$$

Then, Algorithm 1 represents the mathematical steps followed to perform the extraction and labelling in order to

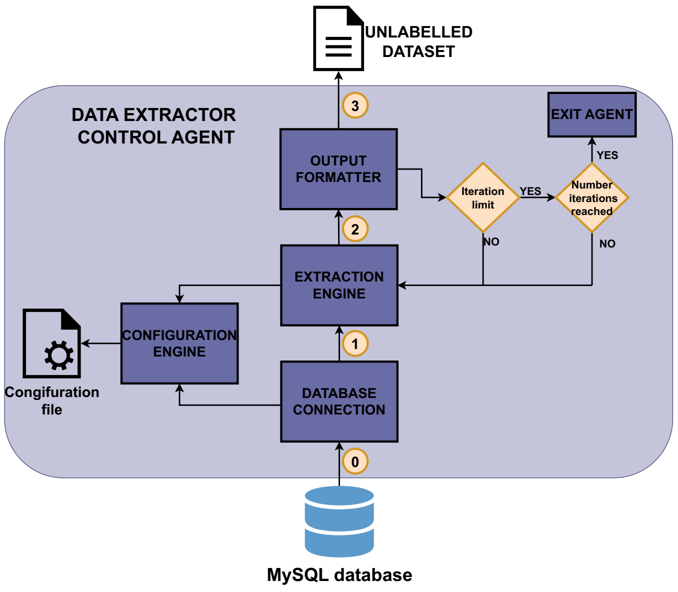


Fig. 4. Data extractor control agent design.

**Algorithm 1** Pseudo-code algorithm for the extraction and labelling

```
1: while $n < number\_extractions$ do
2:     for $R$ in available_R do
3:         extract $I_t^{pR}$
4:         human assign target $L_t = I_t^{pR}$
5:     end for
6:     $n \leftarrow n + 1$
7: end while
8: $\mathbb{I} = \{I_1, I_2, ...I_z\}$ for a given $R$ over a time $z$
9: $\mathbb{L} = \{L_1, L_2, ...T_z\}$ for a given $R$ over a time $z$
```

obtain the label dataset. Finally, note that $\mathbb{I}$ corresponds to the unlabelled dataset, while $\mathbb{L}$ to the labelled dataset.

As specified in line 4 of Algorithm 1, the time it takes to label the data is human-dependent. Therefore, this algorithm is always determined by the time it takes for the human to respond to the system. Assuming line 4 would be automatic, Algorithm 1 exhibits a time complexity of $O(n^2)$ in Big O notation, as there are two nested loops iterating The first loop iterates n times, representing the number of iterations, and for each of these iterations, a second loop is executed, iterating r times, which represents the number of rules being monitored.

## B. Data Extractor Control Agent

Fig. 4 shows the architectural design of the DECA component. The DECA is responsible for data extraction and data adequacy. Once the component is started, the following workflow proceeds. First, the database connection is made (see 0 in Fig. 4). Then, once the connection has been successfully made, it is time for data extraction. Note that the data reported by the NetLabeller agents have already been populated in the database by means of the data collector.

The DECA extraction engine (see 1 in Fig. 4) carries out several tasks. First, it makes a database query to obtain the topology of the network. Then, it performs in parallel the necessary queries to obtain all the features previously described in Section IV. One query to obtain the device information, another one to know the features of the device port and its datapath points. The third query is to obtain the flow features and, finally, the fourth one to obtain the metrics related to the rules currently enforced on the network. In addition, it connects to the configuration engine to store information related to whether the extraction has a finite number of instances to be extracted and, if so, how many to extract. It is also responsible for monitoring a counter to keep track of the number of instances extracted so far. When the extraction is finished, the output formatter module starts (see 2 in Fig. 4). It oversees writing all the features corresponding to an instance in each of the CSV files. Once the writing has finished, it is checked whether it was set to have finite iteration instances. If false, the extraction engine starts again, adding another instance to the counter. If true, the value of the counter and the number of iterations are compared. If it is the same, the extraction has finished and the exit agent stops the component. If not, data extraction continues. The result of the DECA is a CSV file containing all the network features, which is represented in Fig. 4 as the unlabelled dataset (see 3 in Fig. 4). This design allows quantifying streaming data in iterations that are used for training purposes. This is a key ingredient in addressing the streaming aspect associated with the dynamics of a fully functional network infrastructure. It also allows the preparation of the AI training aspect that works more on batches or iterations.

Table III displays a sample of one instance of the dataset generated by the DECA. Note that features are the ones described in Section IV. It is also important to highlight that, for simplicity, only the metrics extracted from the TC datapath technology are being shown. Therefore, in a network interface where there are also e.g., IPTABLES and OVS, the number of features associated with the datapath would multiply.

Once the DECA has finished, the unlabelled dataset is ready for the next step, to be labelled. With Table III, reader can gain a profound understanding of the complexity involved in comprehending and labelling such a vast amount of data in the absence of a visualisation tool. It is therefore necessary to use a visualisation and labelling tool such as NetLabeller.

## C. NetLabeller: Data Visualisation and Labeller Agent

NetLabeller is the labelling tool in charge of labelling the dataset. Fig. 5 shows the NetLabeller design. NetLabeller is a graphical user interface (GUI) that allows the user to visually interact with a networking dataset and to direct label it. NetLabeller abstracts all the complexity that a user can find in a CSV file, as it transforms a bunch of values and commas into an interface that represents all those values in an orderly and visually arranged way. As it is shown in the figure, the design follows a model-view-controller (MVC) pattern. This pattern is based on the separation of the application domain model and the user interface in two layers, having the domain model being unaware of the user interface [35].

Once the application is started, the workflow follows the steps described below. First, the configuration file is uploaded

TABLE III
SAMPLE OF FEATURES EXTRACTED BY DECA IN A CONCRETE ITERATION.

| Feature name | Value | Feature name | Value |
|---|---|---|---|
| Is physical | 0 | Packet size | 284 |
| Context switches | 9 | Total pkt | 8163 |
| Device port speed | $10^9$ | TC RX pkt/sec | 365 |
| TC NIC speed | $10^9$ | TC RX bytes/sec | 118260 |
| Time search complex | 8 | TC RX pkts drop/sec | 269 |
| TC number queues | 1 | TC TX pkt/sec | 0 |
| TC queue length | 1000 | TC TX bytes/sec | 0 |
| TC queue discipline | $noqueue$ | TC TX pkts drop/sec | 0 |
| TC queue max bwd | – | Total matched pkts | 109 |
| Rule technology | $TC$ | Current matched pkts | 109 |
| TC maximum rules | 4096 | Avg matched pkts | 109 |
| Num rules activated | 5 | Last matched time | 0 |
| Rule description | $InsertDROP$ | Total matched bytes | 35316 |
| Rule sense | $INGRESS$ | Current matched bytes | 35316 |
| Rule complexity | 3 | Avg matched bytes | 35316 |
| Encapsulation | $gtp$ | Activated rule time | 0 |
| Flow sense | $INGRESS$ | TC queue usage | – |

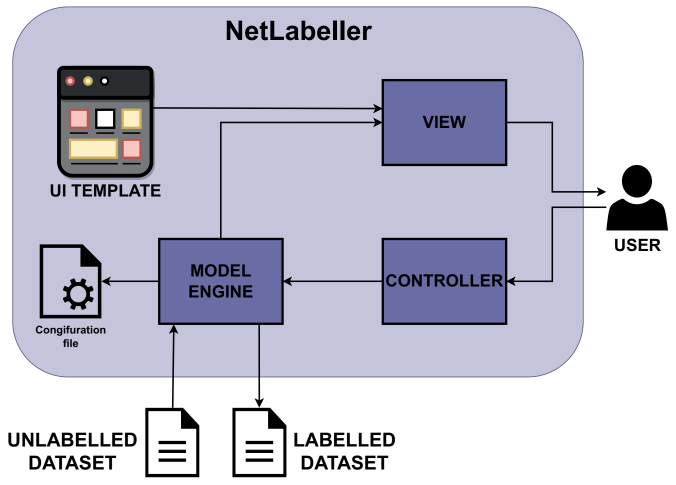


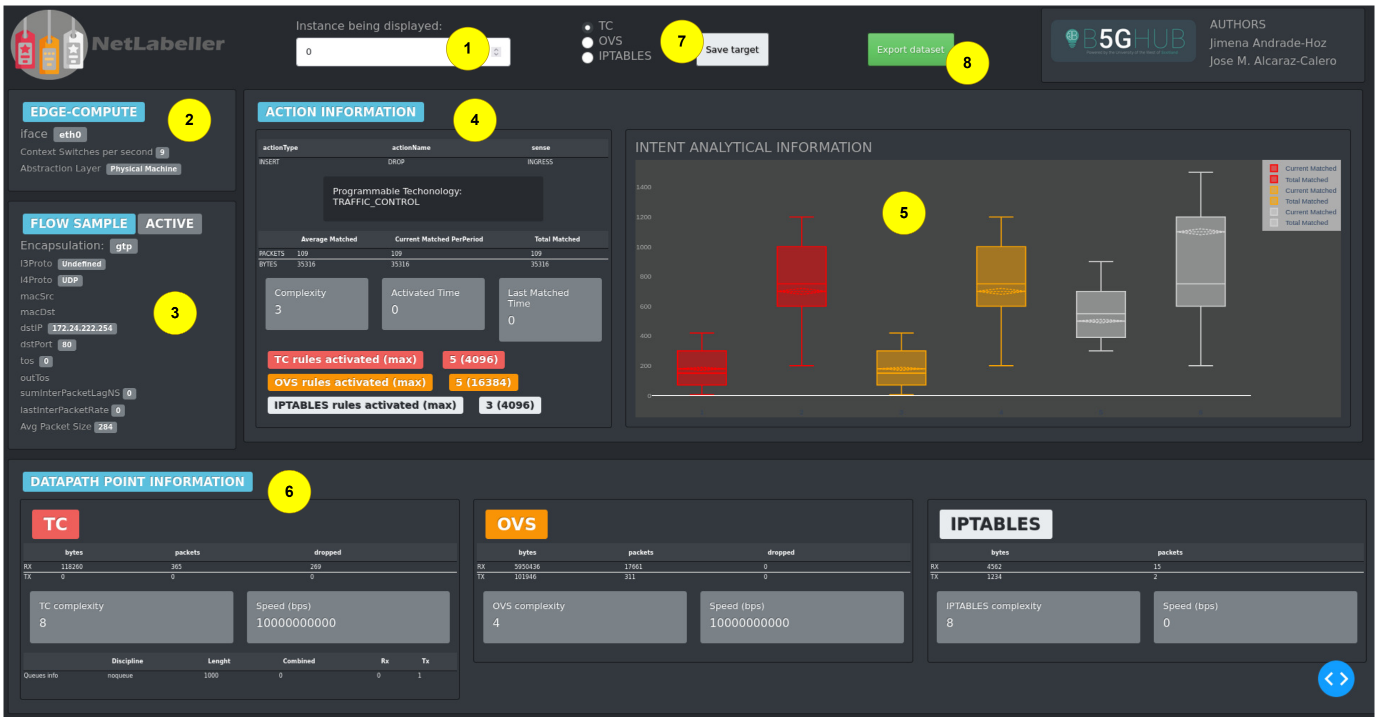
Fig. 5. NetLabeller labelling tool design.

to the model engine. The configuration file has the information about what CSV file to upload, among others. This CSV file is called Unlabelled Dataset in the figure. The model engine loads all the values contained in the dataset at once and sends the first iteration to the view module to be displayed in the user interface (UI). In the meanwhile, the view module loads the UI template in which the data from the model will be displayed. Having the template and the data to be displayed, the view module renders the GUI and displays it to the user. The view module allows the user to (i) select the instance to be displayed, (ii) select the value to put in the label and (iii) save the dataset. The controller oversees the handling user inputs. On certain user actions, the controller triggers model methods that will result in (i) changing the instance and displaying the new values, (ii) saving the value selected and moving automatically to the next instance or (iii) grouping all the values stored and create a new dataset which contains both the unlabelled dataset values and the labels. This dataset is represented in the figure as the labelled dataset.

The NetLabeller GUI is presented as follows. Fig. 6 shows the NetLabeller GUI once it has been run. Before starting the application, it is necessary to edit the configuration file to select the dataset to be labelled. Once the configuration is finished, the Python component can be started. The data presented in Table III have been used to show their representation in the NetLabeller tool. Different yellow circles have been added to the figure for a better explanation of the user interface. In number 1 it is displayed the actual instance whose data is being visualised in the dashboard. The user can select other instances by simply clicking on the input box. On the cards selected with number 2 and 3, both device and flows metadata and metrics are displayed. In section number 4 is displayed the rule information where it can be shown the rule action type and name, the rule technology, or the different rule metrics. In the same section, marked with the number 5, some boxplots are represented to allow the graphical and exploratory analysis of the metrics gathered separated by mean, quarterlies, standard deviations and a descriptive quantifier. In summary, the graph illustrates a descriptive analysis of all the rules applied to the network, providing a better understanding of the rule's performance. These boxplots allow to make an in-depth exploratory data analysis of the feature metrics extracted. Specifically, the values corresponding to the currently matched packages are represented in red and the total matched packages in TC, respectively. The same has been done with OVS, in orange, and IPTABLES, in white. The information about every boxplot can be expanded by simply clicking each of them. Section VI displays metrics associated with the available datapath technologies: TC, OVS and IPTABLES in our testbed infrastructure. The user can select the label to add in the checkbox marked in number 7 and also save it. Finally, once the dataset is completely labelled, it can be saved with the save as CSV button marked in number 8. NetLabeller will generate a new CSV with all labelled instances. The name of this dataset must also be specified in the configuration file.

The tool allows resuming an unfinished labelling task and other user-centric features to achieve an efficient data labelling process. It is worth mentioning that this dashboard has been on-boarded as a plugin into the NetLabeller and the tool allows to customise them for the concrete labelling intention foreseen by the user.

The exploratory data analysis is achieved by using the GUI. The EDA process allows the user to understand the data and gather as many insights as possible from it. Making use of it, the user performs critical investigations on data that will result in labelling such data. The GUI template has been designed to represent all the features of one iteration in the same dashboard. More specifically, the characteristics of each technology have been grouped and represented in different colours. This allows the user to perform a quicker exploratory analysis by associating the different technologies with different colours. It is also worth mentioning that this way of representing all the technologies at a glance, close to each other and all the contextual information about the technologies in the same dashboard allows an intrinsic horizontal exploration of the data. Furthermore, the capability to jump to any iteration



Instance being displayed: 0

AUTHORS
Jimena Andrade-Hoz
Jose M. Alcaraz-Calero

EDGE-COMPUTE
iface eth0
Context Switches per second 9
Abstraction Layer Physical Machine

FLOW SAMPLE ACTIVE
Encapsulation: gtp
l3Proto Undefined
l4Proto UDP
macSrc
macDst
dstIP 172.24.222.254
dstPort 80
tos 0
outTos
sumInterPacketLagNS 0
lastInterPacketRate 0
Avg Packet Size 284

ACTION INFORMATION

| actionType | actionName | sense |
|---|---|---|
| INSERT | DROP | INGRESS |

Programmable Techonology: TRAFFIC_CONTROL

| | Average Matched | Current Matched PerPeriod | Total Matched |
|---|---|---|---|
| PACKETS | 109 | 109 | 109 |
| BYTES | 35316 | 35316 | 35316 |

Complexity 3
Activated Time 0
Last Matched Time 0

TC rules activated (max) 5 (4096)
OVS rules activated (max) 5 (16384)
IPTABLES rules activated (max) 3 (4096)

DATAPATH POINT INFORMATION

TC

| | bytes | packets | dropped |
|---|---|---|---|
| RX | 118260 | 365 | 269 |
| TX | 0 | 0 | 0 |

TC complexity 8
Speed (bps) 10000000000

| | Discipline | Lenght | Combined | Rx | Tx |
|---|---|---|---|---|---|
| Queues info | noqueue | 1000 | 0 | 0 | 1 |

OVS

| | bytes | packets | dropped |
|---|---|---|---|
| RX | 5950436 | 17661 | 0 |
| TX | 101946 | 311 | 0 |

OVS complexity 4
Speed (bps) 10000000000

IPTABLES

| | bytes | packets |
|---|---|---|
| RX | 4562 | 15 |
| TX | 1234 | 2 |

IPTABLES complexity 8
Speed (bps) 0

Fig. 6. NetLabeller GUI for visualising and labelling networking datasets.

.

available in the dataset allows a vertical exploration of it in order to establish comparisons among two similar scenarios.

## D. Sequence Diagram

Fig. 7 shows a sequence diagram representing the detailed communications between the components of the management architecture. Such interactions make the operation of the DECA possible. To achieve a better figure explanation, different numbered labels have been included in the figure. The numbers represent interactions between DECA and other network agents. The rectangles at the top of the figure represent all the agents involved. The interactions are time-dependent, so they happen in the order specified in the figure, from the top down.

The first step (see label 1 in Fig. 7) involves the DECA being started and connected to the database. For the connection, it is necessary to extract the database credentials from the configuration file. For this reason, the database connection module connects with the configuration engine, where the credentials are obtained and returned to the database connection module (see label 2). Then, all the components represented in Fig. 7 should be started and running. To reduce the size of the diagram, it is assumed that the components have been previously started up and are already in operation.

The second step is made by RIA (see label 3). RIA publishes into RabbitMQ the discovered topology in a JSON file. Every component subscribed to the topology exchange in RabbitMQ will receive such information. These are the RMA, the FCA and the data collector. Once the topology is published, different actions are carried out in parallel before the DECA starts its loop. They have been described in a certain order but executed simultaneously.

- Looking at label 4, the data collector transforms the information related to the topology into a SQL sentence and writes the information into the corresponding table.
- In the meantime, looking at label 5, the RMA uses this topological information to run the scripts that will output the related metrics associated to the devices and device ports available in such topological information and publishes them. It is important to note that, for the sake of simplicity, the diagram omits the steps of the components themselves, focusing only on the exchanges of information between them.
- Focusing on label 6, the data collector is also subscribed to the metrics exchange. For this reason, once the metrics form the RMA have been published, it can transform them into a SQL sentence and store this information in the corresponding table.
- Additionally, the FCA uses the topology to know where it can apply rules and publish the associated rule metrics. Label 7 is inside an optional box (opt) as the FCA will only publish rule metrics if there are rules enforced in the network.
- Finally, as shown in label 8, the data collector does the appropriate with the metrics associated with the rules enforced on the network.

At this point, the DECA performs the data extraction loop (see loop box in Fig. 7). That loop gets all the topological information of the network (see label 9). Then, as many parallel query threads are created as device ports with enforced rules are on the network (see the parallel box in label 10). In

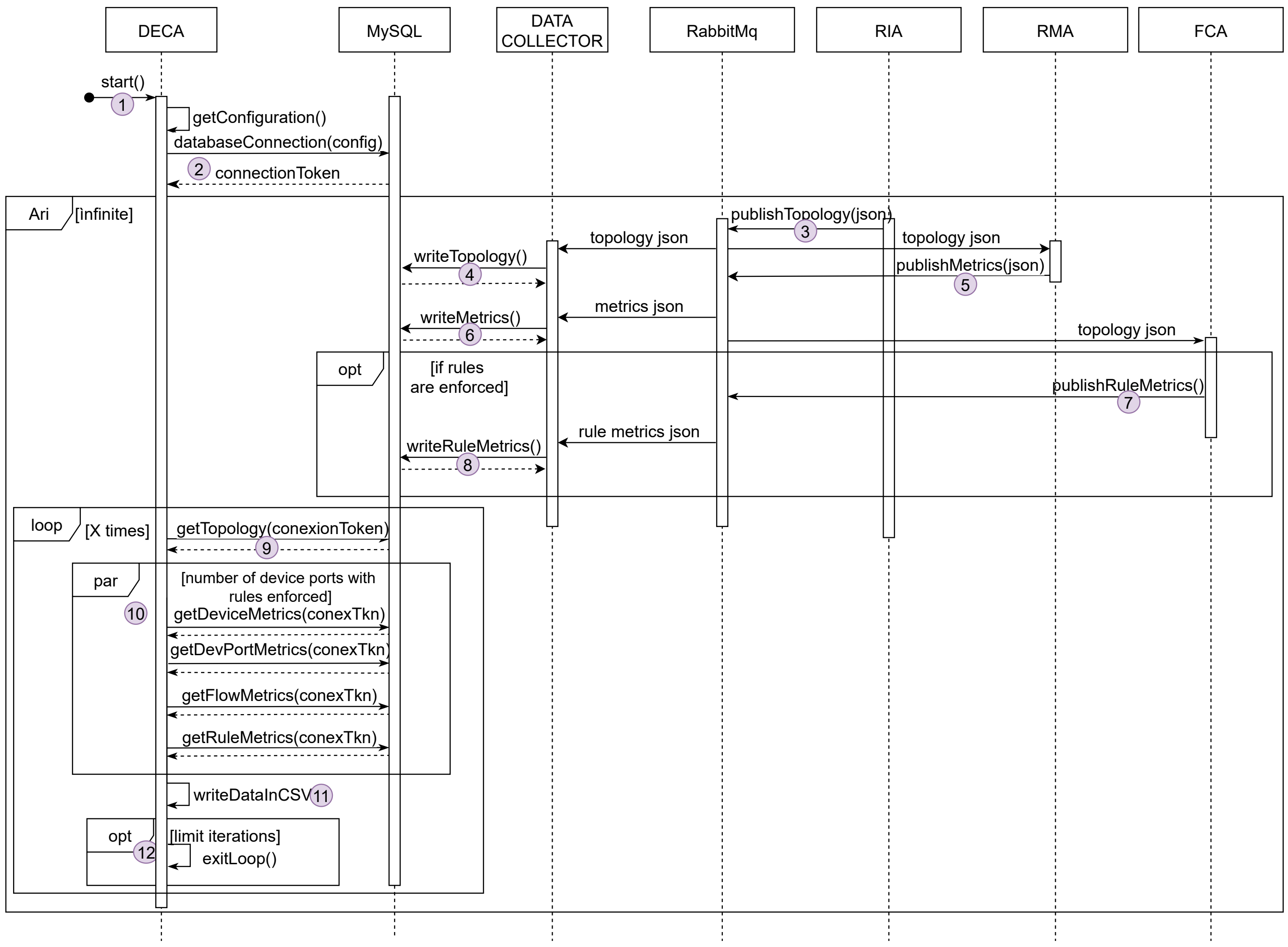


Fig. 7. Sequence diagram representing the connections between components necessary for DECA to extract data.

each thread, a total of four queries are made to obtain the information related with devices, device ports, flows and rules enforced. Then, the information extracted is stored in a CSV file (see label 11) and optionally, in label 12, the DECA is stopped if there is a limitation in the number of iterations to extract.

## VI. Implementation Details and Testbed

This section is divided in two sub-sections. First, it provides some implementation details of the fully functional prototype achieved of the infrastructure. Then, information about the testbed infrastructure deployed in order to validate and evaluate the proposed framework is given. A detailed explanation of the hardware used to deploy the framework prototyped is also provided.

### *A. Implementation Details*

The complete NetLabeller framework has been designed and prototyped. In terms of software specifications, the components RIA, RMA and the data collector are implemented in Java openjdk version 17.0.3 and FCA in Python 3.7. The software used for the implementation of both components, DECA and NetLabeller, is Python 3.8.10. The DECA component basically uses libraries that allow data manipulation and analysis, such as Pandas [36] and NumPy [37]. NetLabeller makes use of libraries related to data visualisation in a GUI and data visualisation in graphs such as Dash [38] and Plotly [39]. Dash allows the deployment of interactive dashboards in a web application. NetLabeller also makes use of Pandas and Numpy for data manipulation. Regarding the softwarised 5G network, the RAN and core of the OpenAirInterface software components (git develop branch) [40] were implemented. Finally, we employ RabbitMq version 3.8.2 as message bus and MySQL 8.15 as database.

### *B. Testbed Infrastructure*

Fig. 8 illustrates the testbed deployed to carry out the empirical validation and evaluation of the NetLabeller framework proposed. Our research group has a 500-core mid-size data centre with a fully operational multi-tenant advance 5G network infrastructure with some prototypical capabilities associated to beyond 5G infrastructures such as O-RAN Near-real-time radio intelligent controller (RIC). The infrastructure is based on Linux Ubuntu 20.04 with Metal-as-a-Service (MaaS)

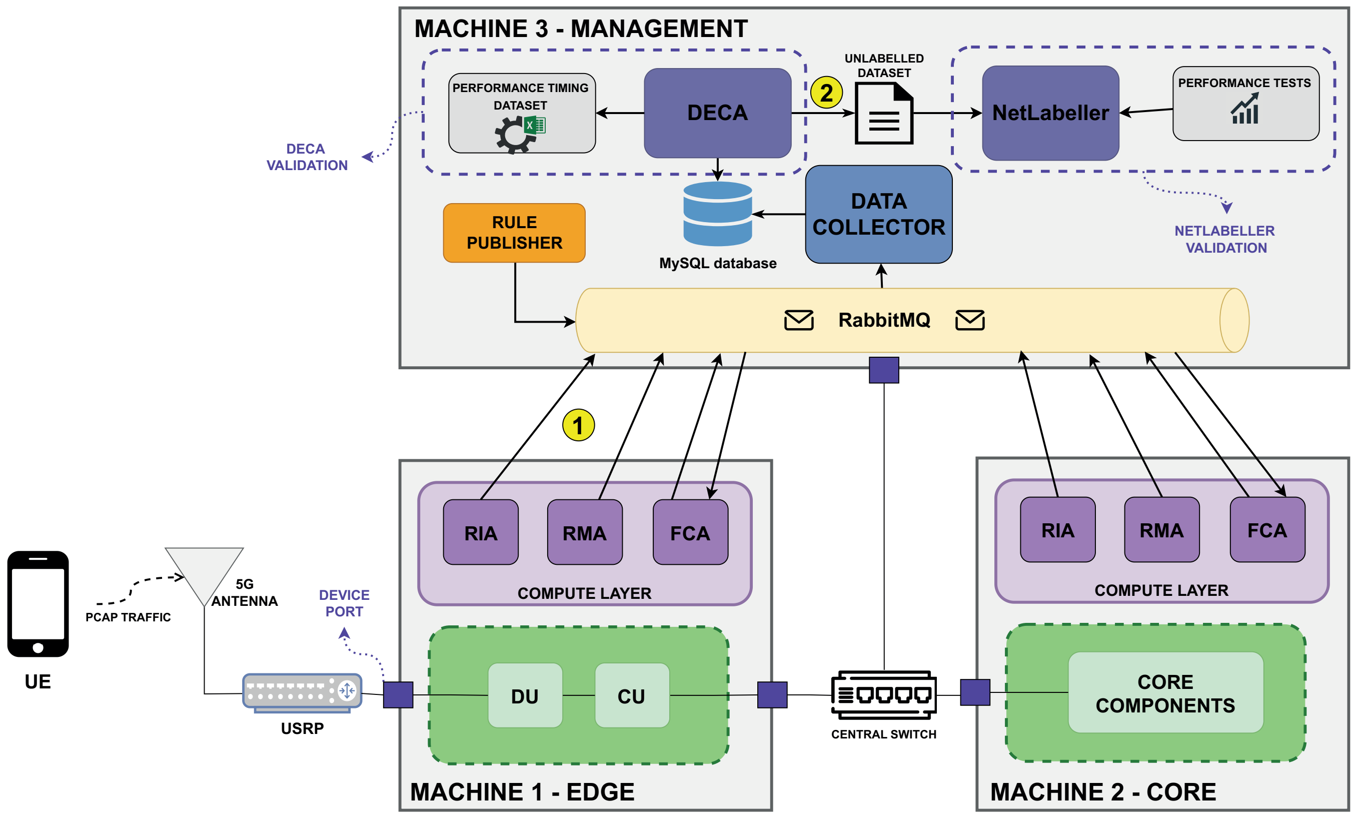


Fig. 8. Architecture and workflow of the testbed deployed for the empirical evaluation of the framework.

3.0, Canonical Juju 1.8, OpenStack Wallaby and OpenAirInterface (OAI) 5G-RAN and Core. Three physical computers have been used for the deployment of the complete infrastructure. Each computer has the following specifications: Dell Precision T5810 with an Intel Xeon CPU E5-2630 v4 CPU, 10 cores with hyper-threading, 32794 MB of RAM and 512 GB SSD hard disk. The operative system is Ubuntu release 20.04.4 LTS. For the radio unit (RU), a universal software radio peripheral (USRP model b210) and a BLUESPOT mini antenna have been used. For simplicity, there is only one user equipment (UE) connected to the RAN and one tenant (represented in green colour on the figure). As shown in the figure, the testbed is composed of three different machines connected to the same LAN. In machine 1 the OAI DU and CU have been deployed. Also, the purple box inside the edge represents the compute layer where NetLabeller agents are deployed. In machine 2, the OAI Core components have been deployed. It can be seen the same compute layer containing the NetLabeller agents deployed in the core machine. Finally, the whole management framework has been deployed in machine 3. It includes the RabbitMq message bus, the data collector, the MySQL database, the DECA and the NetLabeller software. Finally, we have deployed a component called Rule Publisher whose purpose is to publish rules in the network. It is a way to insert rules in a programmatic way, thus impersonating a user administrator to insert a control rule in the network. It was develop for conducting the following experiments.

## VII. System Validation and Performance

This section presents the NetLabeller framework validation and performance. In subsection VII-A, the methodology adopted to execute the different experiments and analyse the results is presented. Then, in subsections VII-B and VII-C, some results showing the validation of the framework are illustrated.

### A. *Experiments Description*

Along the complete execution of the experiments, the UE is simply sending a packet capture (PCAP) file to machine 2 using the Linux command `tcpreplay`. Note that the experiments have been carried out once the UE is connected to the network. The traffic sent by the UE reaches the edge (machine 1 in Fig. 8) and the core through the central switch. While traffic is passing through the network, the NetLabeller agents in the compute layer are each performing their specific function. Therefore, the RIA is publishing the topology information and the RMA, metrics associated with the machine resources. Whereas the FCA is both publishing rule metrics and listening for directives to enforce new rules.

In order to initiate the extraction process, the DECA relies on pre-existing rules that are already enforced on the network. To fulfil this requirement and only for experiment purposes, the Rule Publisher has been utilised. Rule Publisher is a Python component that publishes in a JSON file the necessary directives for the FCA to enforce rules in the network. It is possible to configure the type of rule, where to apply it, the technology to use and the number of rules to publish. In our testbed, it will be possible to apply rules to any of the three network device ports that machines 1 and 2 have (see label of device port in Fig. 8). In addition, the FCA is implemented to support rules from TC, OVS and IPTABLES technologies, so the experiments will be done with rules from these three technologies. The data collector is subscribed to the RabbitMq message bus and transforms all the messages into SQL queries.

Once all the components are running and there are some rules enforced on the network, the DECA is started. It will extract the features a pre-set number of times. The extraction is performed every pre-determined number of seconds. These parameters have been set in the configuration file. At each iteration, in addition to writing the metrics extracted from the network to the unlabelled dataset, DECA writes in a CSV instrumentation file the times taken to complete every step labelled in Fig. 4 with yellow circles. As shown in Fig. 8, this data will be analysed later in a Microsoft Excel file. Finally, the CSV generated by the DECA will be uploaded to the NetLabeller. Some performance tests have been carried out during the labelling process in order to analytically measure the performance of the tool. A more detailed description of these experiments is provided in the following sub-sections.

Concerning the procedure followed to execute the experiments and gather results, all the experiments have been reproduced five times. To achieve more accurate results, the arithmetic mean of these values has been done. As the performance of two very different tools will be tested, the procedures for each will be described in each of the sub-sections. In subsection B, some experiments regarding the DECA performance have been carried out. In subsection VII-C, focusing on the NetLabeller tool, an explanation of some performance tests made in the web application are presented. The performance tests that have been carried out on NetLabeller analyse the quality of the user experience in terms of navigation time and other performance-related times. Finally, in sub-section D, the overall performance of the proposed framework. Such performance has been measured in terms of latency. This performance analysis has been carried out by measuring latency times throughout the storage, extraction and labelling process.

## B. *Data Extractor Control Agent Performance Analysis*

To perform the DECA experiments, times have been collected in four different parts of the component code and have been written in an Excel file. These steps are indicated in Fig. 4 in yellow circles. Step 0 registers the time when the DECA is started. Step 1 saves the timestamp when the connection to the database has been successfully established. Step 2 registers when the feature extraction has finished and step 3 saves the timestamp when the instances have been written in the different CSV files. The differences between steps represent how much does the DECA take to perform the three modules depicted in Fig. 4: i) Database connection, ii) data extraction and data adequation and iii) data output witting. All experiments have been executed during 100 iterations.

As described previously, the DECA extracts data every number of seconds chosen in the configuration file. The first experiment has been carried out to check what is the minimum

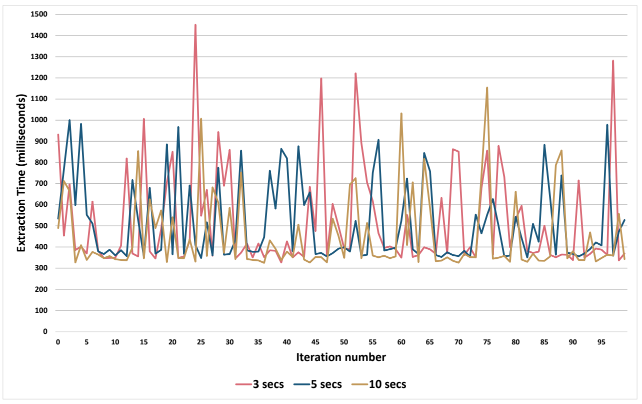


Fig. 9. Evolution of the execution time of DECA along hundred different extractions.

time the component needs to finish one iteration and what is its performance depending on the number of seconds defined. To do so, three different times have been chosen: 3 seconds, 5 seconds and 10 seconds. The test scenario is the same for all three cases, with the same number of rules in the network and the same traffic sent from the UE. Five repetitions have been performed for each time between iterations (3, 5 and 10 sec), then averaging these values to obtain more accurate results. Fig. 9 represents the evolution of the total time taken by DECA to complete an extraction of each iteration. It can be seen that the time is very variable, although it should be noted that it is a matter of milliseconds of difference. This variation occurs in all three cases analysed. The results show that, in the worst case, DECA takes 1.45 seconds to finish the iteration. This worst case corresponds to the experiment when 3 seconds per extraction are selected. Fig. 10 represents the total average time taken by DECA, over these 100 iterations. The total execution time is shorter when iterations are performed every 10 seconds. Again, it should be noted that these differences are milliseconds. The standard deviation has also been represented in the figure. It can be noted that it is significant in relation to the mean, which indicates that data are spread out. This measure confirms that the results are variable, as shown in Fig. 9. Apart from this fact, the time in all cases is around half a second. This allows the reader to understand what minimum response time is required to generate the dataset. Such time is directly related to the minimum time it will be required to update the AI model during the training phase.

The second experiment has been carried out to prove the DECA scalability, as performance may vary depending on the number of rules in the network. For this experiment, a fixed time between iterations has been chosen, which is 10 seconds as proved before that is the optimum value. Then, results from a variant number of rules have been gathered: 1 to 32 in steps of 2 rules (1, 2, 4, $\cdots$, 30, 32) enforced in the network. Again, five repetitions have been executed for each number of rules. The results shown in Fig. 11 are the arithmetic average of the repetitions. The graph represents a bar per number of enforced rules. Each bar has 3 different colours, representing the times in milliseconds the DECA takes for each step. As we can see, the database connection time is

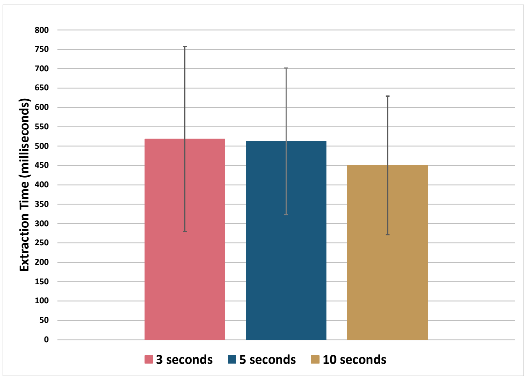


Fig. 10. Results of the average DECA execution time for extraction intervals. Mean and std dev. of the plots in Fig. 9.

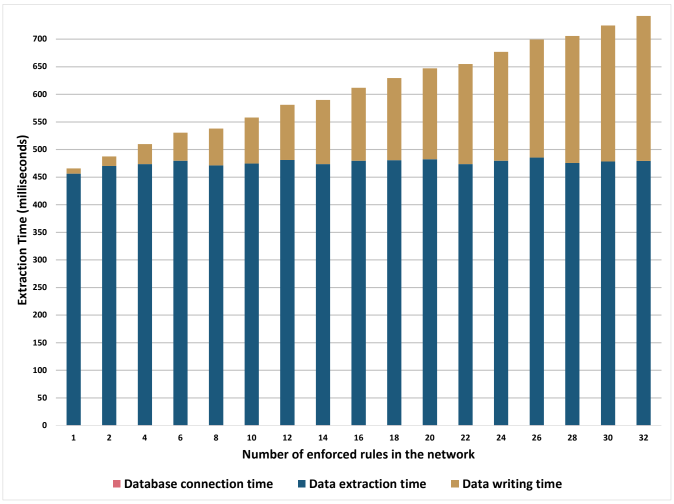


Fig. 11. Results of the average execution time of each of the DECA steps as a function of the number of rules enforced in the network.

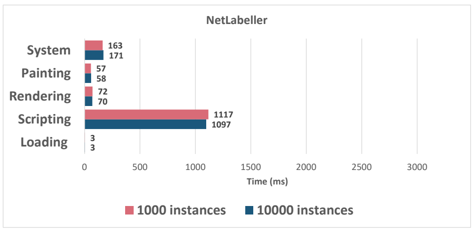


Fig. 12. NetLabeller runtime performance analysis.

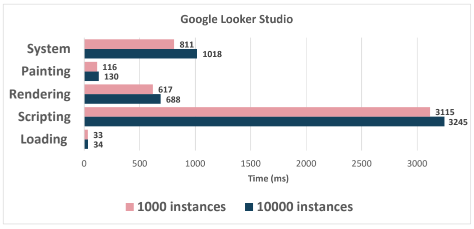


Fig. 13. Google Looker Studio runtime performance analysis.

insignificant in comparison with the rest, with an average of 0.5 milliseconds. The data extraction time is depicted in blue. Is the most time-consuming of all, for a duration of almost half a second (around 470 milliseconds). It is worth noting that the extraction time does not depend on the number of rules enforced in the network, as it is a process that is done in parallel by several threads. Therefore, the times are practically the same in all tests. Finally, we can see the data writing time in mustard-coloured. It is observed that the writing time increases linearly with the number of rules by steps of 16 milliseconds approximately. This makes sense since a different CSV is written for each rule enforced in the network. This experiment proves that the average extraction time is less than 700 milliseconds in most cases and independent of the number of network rules.

### *C. NetLabeller Runtime Performance Analysis*

Runtime performance shows how the NetLabeller web page performs when it is running. Chrome DevTools Performance panel [41] has been used for analysing the NetLabeller performance. The panel allows to record the page and provides a detailed explanation of the actions being executed. The explanation includes the following times about the page performance: Loading, scripting, rendering, painting, system and idle. For our analysis, the idle time has been ignored, as can be considered as the time taken by the user to think about what label to put according to the data being displayed. This time will depend on the user; therefore, it is of no interest when measuring the performance of the tool. Performance evaluation consisted of measuring these times, varying the size of the dataset to be labelled. That is, by changing the number of instances. For this purpose, the following actions have been carried out in all cases:

1) Launch the framework with X instances.
2) Start recording the performance.
3) Reload the page.
4) Select a technology and click on the save button.
5) Repeat step (4) 5 times.
6) Click the save button to save the CSV file.
7) Stop the recording.

The X number in step 1 has been ranged with values 10, 100, 1000, 10000 and 100000 instances respectively. These actions have been repeated 5 times for each number of instances. The arithmetic means of the results obtained were then calculated. Fig. 12 shows bar charts of the results obtained with 1000 and 10000 instances. As the reader can see, both cases show a very similar distribution of the times. This means that the NetLabeller runtime performance is not affected by the number of instances to be labelled. This is a very desirable feature of our framework, as it means that we will not have performance problems if the number of instances to label is very high. These results are similar to the cases not plotted for a shake of simplicity (10, 100 and 100000).

To highlight the excellent performance of our framework, we have performed a comparison against one of the visualisation tools presented in Table I. Google Looker Studio has been used for this purpose because of its user-friendly and intuitive interface. We produced a similar dashboard, where we present the same features as those shown in Fig. 6. We then followed the same steps as above. Except for step 6, since as mentioned in Section II, this tool does not allow CSV labelling. Fig. 13

shows the obtained results. They show two clear conclusions: (i) The NetLabeller's runtime performance is better in any of the 5 measured times and (ii) the performance of second tool used is affected by the number of instances to be labelled, in contrast to ours.

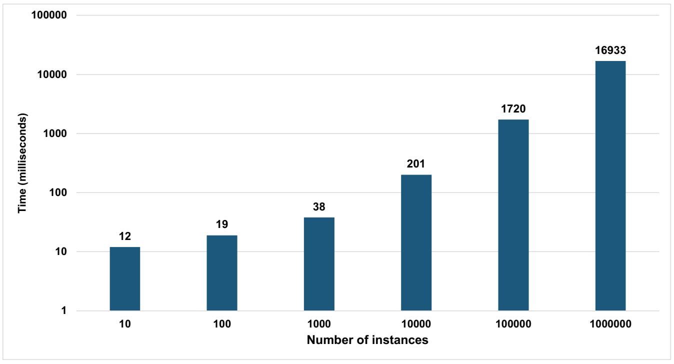


Fig. 14. Time taken by NetLabeller to save the labelled dataset, depending on the size of the input dataset.

On the other side, at a network level, the analysis of the runtime performance shows a clear linear relationship between the number of instances and the time NetLabeller takes to save the labelled dataset. The relationship can be shown in Fig. 14. Notice the time is on a millisecond scale which clearly validates the good performance of the tool as it takes 16 seconds in recording 1 million iterations in labelled data. Finally, it is worth noting that the dashboard updates instantaneously when switching from instance to label. This is because NetLabeller is a SPA (single-page application), so the data is loaded all at once when the application is started. This makes navigation through the application highly fluid.

### *D. Validation of Feature Completeness*

The methodology employed in the design of the dataset features entails incorporating all feasible metrics and metadata accessible within the system pertaining to the decision-making procedure. Through the adoption of this complete enumeration approach, the dataset guarantees the provision of exhaustive information for the network expert involved in the decision-making process. It is noteworthy to emphasise that the dataset features include all the key elements involved: Device, device ports, datapath technologies, datapath queues, network flows and network control rules. The complete enumeration approach is commonly used in scenarios where it is feasible to consider all variables and their possible impacts. It is also employed as a means of addressing situations where uncertainty exits about the predominant variables, as is the case in this particular research. Such problem-solving approach has also been used for other network-related problems, for example by Koide *et al.* in [42].

In order to validate it, a set of 10 network security, network engineers and network managers available in our computing department have been using the NetLabeller framework to determine if the information presented is enough to take the decision on the best technology to be used. There is a unanimous 100% agreement among the participants that the information is enough for the decision-making process which is a clear insight of the completeness of the dataset design. However, what is not unanimous (in at least 20% of the scenarios, mainly those that are not clear cases) is what technology they have chosen as the optimal among the 50 scenarios being analysed. This confirms our original idea related to the need of AI that help network experts to perform optimal decisions, based on the given current state of the system.

## VIII. Conclusion

This paper has presented a novel architecture to extract and label networking data coming from beyond 5G and potentially future 6G networks. The in-depth study of the state of the art regarding labelling tools has revealed the importance of the contribution presented in this paper, as there is no tool that offers the capabilities that were sought. The proposed architecture has been implemented in a lab-based realistic 5G infrastructure, showing how our contribution will fit in such a real network infrastructure environment. It is composed of two components: the Data Extractor Control Agent, which is capable of extracting network features and writing them in a CSV file in less than a half millisecond; and the NetLabeller, a tool for visualising and labelling the network datasets created. It is our intention in future work to develop a deeper analysis of the metrics presented in this research work, providing further insight into which metrics are most relevant to the decision-taking process. Future work will also further explore and evaluate the applicability of the proposed architecture through the creation and implementation of an AI model for network optimisation use cases. This AI model will use the datasets extracted and labelled by the proposed NetLabeller framework. Furthermore, it is our objective to publish additional research publications, which will include generated datasets. This endeavour aims to allow the research community to have these valuable assets for further research purposes and to validate it.

## References

[1] P. Mitra *et al.*, "Towards 6G communications: Architecture, challenges, and future directions," in *Proc. ICCCNT*, 2021.
[2] T. Huang *et al.*, "A survey on green 6G network: Architecture and technologies," *IEEE Access*, vol. 7, pp. 175758–175768, 2019.
[3] H. Alquhayz, N. Alalwan, A. I. Alzahrani, A. H. Al-Bayatti, and M. S. Sharif, "Policy-based security management system for 5G heterogeneous networks," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–14, 2019.
[4] M. Asim *et al.*, "Security policy monitoring of BPMN-based service compositions," *J. Software: Evol. Process*, vol. 30, no. 9, p. e1944, 2018.
[5] G. N. Purdy, *Linux iptables Pocket Reference: Firewalls, NAT & Accounting*. "O'Reilly Media, Inc.", 2004.
[6] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1273–1286, 2016.
[7] G. Choudhury, "XDP-programmable data path in the Linux kernel," *Spring*, vol. 43, no. 1, 2018.
[8] B. Hubert *et al.*, "Linux advanced routing & traffic control," in *Ottawa Linux Symposium*, vol. 213, sn, 2002.
[9] A. Kaloxylos, A. Gavras, D. Camps, M. Ghoraishi, and H. Hrasnica, "AI and ML–enablers for beyond 5G networks," 5G PPP, Tech. Rep., 2021.

[10] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84–90, 2019.

[11] H. Huang, J. Chu, and X. Cheng, "Trend analysis and countermeasure research of DDoS attack under 5G network," in *Proc. IEEE CSP*, 2021.

[12] P. Salva-Garcia, R. Ricart-Sanchez, E. Chirivella-Perez, Q. Wang, and J. M. Alcaraz-Calero, "XDP-based smartnic hardware performance acceleration for next-generation networks," *J. Netw. Syst. Manag.*, vol. 30, no. 4, p. 75, 2022.

[13] B. Pfaff *et al.*, "The design and implementation of Open vSwitch," in *Proc. NSDI*, 2015.

[14] N. Baldo, L. Giupponi, and J. Mangues-Bafalluy, "Big data empowered self organized networks," in *Proc. European Wireless Conf.*, 2014.

[15] L. Stopar, P. Skraba, M. Grobelnik, and D. Mladenic, "StreamStory: Exploring multivariate time series on multiple scales," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 4, pp. 1788–1802, 2019.

[16] N. Y. Philip *et al.*, "A data analytics suite for exploratory predictive, and visual analysis of type 2 diabetes," *IEEE Access*, vol. 10, pp. 13460–13471, 2022.

[17] J. DSouza and S. Velan S., "Using exploratory data analysis for generating inferences on the correlation of COVID-19 cases," in *Proc. ICCCNT*, 2020.

[18] Colabeler. *Colabeler: AI labeling tool* [Online]. Available: http://www.colabeler.com/

[19] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov. (2020–2022). *Label Studio: Data Labeling Software.* [Online]. Available: https://github.com/heartexlabs/label-studio

[20] vmware. (2020–2022). *Data Annotator for Machine Learning (DAML)* [Online]. Available: https://github.com/vmware/data-annotator-for-machine-learning

[21] P. Estrada, A. Sarkis, and V. Bulyzhyn. (2020–2022). *Diffgram: Open Source Data labeling Platform* [Online]. Available: https://github.com/heartexlabs/label-studio

[22] M. Chakraborty and A. P. Kundan, "Grafana," *Monitoring Cloud-Native Appl.*, pp. 187–240, 2021.

[23] V. Sharma, "Getting started with Kibana," *Beginning Elastic Stack,* pp. 29–44, 2016.

[24] G. Snipes, "Google data studio," *J. Librarianship Scholarly Commun.*, vol. 6, no. 1, 2018.

[25] M. Chegini *et al.*, "Interactive labelling of a multivariate dataset for supervised machine learning using linked visualisations, clustering, and active learning," *Visual Inform.*, vol. 3, no. 1, pp. 9–17, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468502X19300178

[26] G. Lee, J. Lee, Y. Kim, and J.-G. Park, "Network flow data re-collecting approach using 5G testbed for labeled dataset," in *Proc. ICACT*, 2021.

[27] J. Lee, H. Kim, C. Park, Y. Kim, and J.-G. Park, "AI-based network security enhancement for 5G industrial Internet of things environments," in *Proc. ICTC*, 2022.

[28] L. Peterson and O. Sunay, "5G mobile networks: A systems approach," *Synthesis Lectures Netw. Syst.*, vol. 1, no. 1, pp. 1–73, 2020.

[29] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[30] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2018.

[31] J. Kim, D. Kim, and S. Choi, "3GPP SA2 architecture and functions for 5G mobile communication system," *ICT Express*, vol. 3, no. 1, pp. 1–8, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240595951730019X

[32] I. Sanchez-Navarro, A. Serrano Mamolar, Q. Wang, and J. M. Alcaraz Calero, "5GTopoNet: Real-time topology discovery and management on 5G multi-tenant networks," *Future Generation Comput. Syst.*, vol. 114, pp. 435–447, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1933122X

[33] A. Matencio Escolar, J. M. Alcaraz-Calero, P. Salva-Garcia, J. B. Bernabe, and Q. Wang, "Adaptive network slicing in multi-tenant 5G IoT networks," *IEEE Access*, vol. 9, pp. 14048–14069, 2021.

[34] H. Srivastava, "Queueing Theory," in *Encyclopedia of Physical Science and Technology*, third edition ed., R. A. Meyers, Ed. New York: Academic Press, 2003, pp. 481–495. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0122274105006323

[35] A. Syromiatnikov and D. Weyns, "A journey through the land of model-view-design patterns," in *Proc. WICSA*, 2014.

[36] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proc. SciPy*, 2010.

[37] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.

[38] P. T. Inc. (2015) Dash Python. Montreal, QC. Accessed on 14.08.2022. [Online]. Available: https://dash.plotly.com/

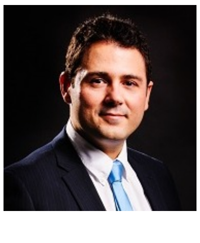
[39] Plotly. (2015) Collaborative Data Dcience. Montreal, QC. Accessed on 14.08.2022. [Online]. Available: https://plot.ly

[40] OpenAirInterface, "OpenAirInterface 5G Radio Access Network Project," Accessed on 10.06.2022. [Online]. Available: https://openairinterface.org/

[41] Google, "Chrome Developers Tool," Accessed on 18.09.2022. [Online]. Available: https://developer.chrome.com/docs/devtools/

[42] T. Koide, S. Shinmori, and H. Ishii, "An efficient complete enumeration method for network design problems and its applications," *J. Operations Research Soc. Japan*, vol. 45, no. 3, pp. 299–316, 2002.

**Jimena Andrade-Hoz** received the B.S. and M.S. degrees in Telecommunications Engineering from the University of Valladolid, Spain, in 2021. She is currently pursuing the Ph.D. degree in the School of Computing, Engineering and Physical Sciences at the University of the West of Scotland. Nowadays she is actively involved in her research project about self-optimisation in autonomous cellular networks. Some of her research focus and interests are: artificial intelligence applied to cellular networks, 5G network management and control, and 5G Networks in Cloud Computing. She is also working in both European Projects: the H2020 ARCADIAN-IoT: Autonomous Trust, Security and Privacy Management Framework for IoT and H2020 RIGOUROUS: secuRe desIGn and deplOyment of trUsthwoRthy cOntinUum computing 6G Services.



**Jose M. Alcaraz-Calero** (Senior Member, IEEE) received the Ph.D. degree in Computer Science from the University of Murcia, Spain. Prof. Jose M. Alcaraz Calero (M) is a Professor in Networks at School of Engineering and Computing at the UWS. He is an IEEE Senior Member with experience in 5G networks, network slicing, monitoring, automation and management. In the academic side, he has more than 150 publications in SCI-indexed international journals, conferences and books. He has been involved in 20 editorial activities in the most prestigious journal in the field and served as chair in 20 international flagship conferences and contributes as Technical Programme Committee member in more than 100 international conferences. In the industrial side, he has more than 50 patents and intellectual property rights. From the leadership perspective, Jose M. has significant experience as Principal Investigator or as Co-Investigator in more than 20 research projects at local, national and especially at European and international level. He is the Co-Technical Manager for EU Horizon 2020 projects SELFNET and SliceNet.

**Qi Wang** received the Ph.D. degree in Mobile Networking from the University of Plymouth, U.K. Prof Qi Wang (M) is a Professor in Next-Generation Smart Networks and Services at UWS. He has served as a Board Member of the EU 5G-PPP Technology Board, Member of several 5G-PPP Working Groups, Member of Scotland's Developing AI and AI Enabled Products and Services Working Group, and the Co-Technical Manager for EU Horizon 2020 projects SELFNET and SliceNet. He is Co-Principal Investigator for EU Horizon 2020 projects 6G BRAINS, 5G INDUCE and ARCADIAN-IoT, and EU Horizon Europe projects RIGOUROUS, INCODE and 6G PATH. He has published about 200 papers in 5G, 6G, AI and related areas. He was a Best Paper Award Winner of IEEE IEEE ICCE2012, ICCE 2014, SIGMAP 2014, SOFTNETWORKING 2017 and PETRA 2023. He was a Winner of 2018 Scottish Knowledge Exchange Award, Winner of Scotland Centre for Engineering Education and Development (CeeD) Industries Awards - Innovation Award 2020, and Winner of UK Times Higher Education (THE) Awards 2020 -Knowledge Exchange/Transfer Initiative of the Year Award.