EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

An experimental publish-subscribe monitoring assessment to Beyond 5G networks

Ramon Perez[1,2*], Jaime Garcia-Reinoso[2], Aitor Zabala[1], Pablo Serrano[2] and Albert Banchs[2,3]

*Correspondence: ramon.perez@telcaria.com
[1] Telcaria Ideas S.L., C/ Barrionuevo, 8, 28911 Leganés, Madrid, Spain
Full list of author information is available at the end of the article

Abstract
The fifth generation (5G) of mobile networks is designed to accommodate different types of use cases, each of them with different and stringent requirements and key performance indicators (KPIs). To support the optimization of the network performance and validation of the KPIs, there exist the necessity of a flexible and efficient monitoring system and capable of realizing multi-site and multi-stakeholder scenarios. Nevertheless, for the evolution from 5G to 6G, the network is envisioned as a user-driven, distributed Cloud computing system where the resource pool is foreseen to integrate the participating users. In this paper, we present a distributed monitoring architecture for Beyond 5G multi-site platforms, where different stakeholders share the resource pool in a distributed environment. Taking advantage of the usage of publish-subscribe mechanisms adapted to the Edge, the developed lightweight monitoring solution can manage large amounts of real-time traffic generated by the applications located in the resource pool. We assess the performance of the implemented paradigm, revealing some interesting insights about the platform, such as the effect caused by the throughput of monitoring data in performance parameters such as the latency and packet loss, or the presence of a saturation effect due to software limitations that impacts in the performance of the system under specific conditions. In the end, the performance evaluation process has confirmed that the monitoring platform suits the requirements of the proposed scenarios, being capable of handling similar workloads in real 5G and Beyond 5G scenarios, then discussing how the architecture could be mapped to these real scenarios.

Keywords: Monitoring, Data collection, 5G experimental validation, 5G trials, 5G multi-site platform, Publish-subscribe paradigm, Beyond 5G networks

1 Introduction

The evolution of mobile networks from 2G to 4G generations was mainly focused on providing a better quality of experience to end users, by increasing the bandwidth offered by the network at the radio link segment. However, 5G networks, together with their expected evolution in what is currently known as Beyond 5G networks, have a broader target, shifting traditional communication networks to a new generation mobile network that embraces other business sectors.

In the case of 5G, the authors of [2] have reported the service requirements expected by verticals, which is the terminology used by 5G to define these business sectors moving to 5G as the main transport infrastructure. Due to the stringent and different requirements imposed by all these potential verticals deploying their services on top of 5G networks, the most important standard development organizations (SDOs) tackling the 5G standardization, like the 3GPP, have introduced the concept of Network Slicing [3], which provides multiple isolated logical networks from a single physical one.

In this approach, each logical network may support a particular type of 5G service, e.g., enhanced mobile broadband (eMBB), massive machine-type communications (mMTC) or ultra-reliable and low latency communications (URLLC). As a matter of fact, 5G telecommunication operators have to design their networks to support all these services and to guarantee that the KPIs demanded by their verticals are satisfied. Beyond 5G networks are also required to enable network deployments that support diverse demands through network slicing.

Triggered by the complexity and novelty of 5G, several research initiatives have started to gather an understanding of the envisioned features of these types of networks. Focusing on the analysis of the achievement of the KPIs aforementioned, a monitoring service is desired for collecting all the related metrics generated by the different elements involved in a 5G scenario, in order to feed such data to specific components that checks the KPIs' fulfillment, or also enabling new workflows like the optimization of network performance.

This paper presents a monitoring framework designed to be flexible enough to provide an adaptable platform that could fit and scale in different network deployments. Apart from covering use cases related to the 5G service types commented before, this platform also intends to anticipate new requirements that Beyond 5G networks may impose, such as efficient resource utilization or real-time traffic processing.

The rest of the paper is organized as follows:

- Section 2 briefly presents some references to related work in terms of monitoring platforms deployed in 5G or Edge scenarios, including the monitoring parameters and methods, and having also standardization in mind.
- Section 3 describes the monitoring architecture, which has been designed as a scalable, reliable, low-latency, distributed, multi-source data aggregation and reconfigurable architecture.
- Section 4 justifies and details an implementation model to deploy the proposed architecture, based on the publish-subscribe paradigm. It also presents and example of how a real 5G use case can be monitored by this platform.
- Section 5 validates such implementation against a set of requirements imposed, also extending the analysis to check the viability of deploying this platform in Edge environments.
- Section 6 summarizes the results and the main conclusions extracted from the performance evaluation process.
- Finally, Sect. 7 concludes the paper and presents our future work.

The main novelty of this paper, compared to the work already presented in [1], is as follows:

- This extension provides a full section covering the related work to this monitoring architecture (Sect. 2), which allows to justify the decision of using the publish-subscribe paradigm for the implementation and validation of the monitoring system presented.
- The implementation of the monitoring platform (Sect. 4) has been reviewed to cover the multi-site scenarios. As most important contributions in this section, (1) the *Data Collection Manager* component is now included within the so-called *Multi-Broker Cluster*, an entity which also contains the different *Site Brokers* deployed per site. Moreover, (2) the information model defining the topics handled by this *Multi-Broker Cluster* has been extended with the distinction between data topics and signaling topics. And finally, (3) some examples of 5G components monitored by this monitoring platform have been also presented in a new subsection (Sect. 4).
- Regarding the performance evaluation process followed in this extension, the testbed used for the test executed has been improved by using *Kubernetes* to easily deploy the Dockerized environment that composes the testbed. Furthermore, the testbed has been updated to enable the evaluation of multi-site, multi-broker scenarios.
- The multi-topic experiments for a single-broker configuration (Sect. 5.3.2) have been extended by including an evaluation of the monitoring parameters under study when limiting the computing resources used. This is summarized in Fig. 8, and it has been included to justify the introduction of this platform in Beyond 5G scenarios.
- A new subsection related to multi-broker experiments (Sect. 5.4) has been included, covering all the tests involving multiple sites.

2 Related work

The monitoring parameters considered as base to build this monitoring platform are the eight parameters proposed by the ITU-R as key capabilities of IMT-2020 [4], which are: peak data rate, user-experienced data rate, latency, mobility, connection density, energy efficiency, spectrum efficiency and area traffic capacity. Apart from these infrastructure-related KPIs, other KPIs related to the use cases deployed in 5G networks can be also considered, which are different for each application considered.

The purpose of the monitoring platform proposed in this paper is to monitor all these infrastructure and application metrics and KPIs with a system based on a publish-subscribe mechanism to collect and distribute the monitoring data through the platform. In this way, as long as the components generating the metrics are able to provide the data to this system, these metrics and their corresponding KPIs can be monitored consequently.

In terms of monitoring methods applied for 5G networks, there is not a specific paradigm defined in standards to be followed in real deployments. Apart from the publish-subscribe mechanism, the push or pull monitoring architectures can also apply to 5G monitoring systems. In the first case, monitoring data are sent to a central collector, and in the second option, this central collector is in charge of requesting the metrics. In both cases, a set of APIs are required to acquire monitoring data or expose it [5].

Moreover, the related work in terms of monitoring platforms designed and provisioned for 5G and Beyond 5G networks can be grouped into three main categories, according to the environment in which the system presented in this paper has been involved.

First of all, (1) this monitoring platform has been designed and implemented within the scope of an European project related to the research on 5G networks: 5G EVE [6, 7]. This project is deploying a validation 5G multi-site platform, involving four main facilities located in Spain, Italy, France and Greece, where verticals and other projects can execute extensive trials. After an initial phase where verticals have provided their requirements (reported in [8]), the project presented in [9] the proposed architecture and the main innovation areas addressed, including the KPI Framework for performance diagnosis.

While a number of other 5G projects (European and International) have addressed monitoring functionalities, limited work in this context has addressed the publish-subscribe paradigm, a messaging pattern which can be commonly found in the communication between distributed systems. This paradigm was the option selected by 5G EVE to implement its monitoring architecture, and this idea was also considered in the 5GROWTH project, integrating some ideas and concepts present in the 5G EVE Monitoring platform the so-called Vertical-oriented Monitoring System (5Gr-VoMS) [10], which is an extension of the monitoring solution already proposed in the 5G-TRANSFORMER project [5].

Another context that is present in these environments (2) is standardization. In order to integrate monitoring and data collection features in the mobile network architecture, 3GPP and other SDOs are working in data analytics frameworks that take advantage of the collection of monitoring data related to the network infrastructure in order to enable the autonomous and efficient control, management and orchestration of mobile networks. In this working line, 3GPP has incorporated the Network Data Analytics Function (NWDAF) [11] and the Management Data Analytics Function (MDAF) [12] for 5G networks.

Other organizations, such as the O-RAN alliance, also contemplate similar components in their architectures [13], and ETSI has also defined comparable assisting elements within the Industry Specification Groups (ISGs) on Experiential Networked Intelligence (ENI) and Zero touch network and Service Management (ZSM) [14]. Moreover, open-source initiatives such as ONAP [15] are also including data analytics into their architecture. All these ongoing efforts are, however, at an early stage, so that the integration of the monitoring architecture presented in this paper, already tested and validated, may be useful to steer the work of these initiatives.

And finally (3) , moving to Beyond 5G networks, requiring flexible scenarios that may be probably oriented to Edge environments, there are already several proposals that include the definition of a publish-subscribe mechanism to distribute data between different entities in Edge-based deployments. This is the case of [16] or [17], although they are mostly focused on IoT and pure Edge platforms, not including 5G communications. There are also other proposals not related to the publish-subscribe system, such as [18], which analyzes the optimal placement and scaling of monitoring functions in multi-access Edge computing (MEC) environments, but it does not consider multi-site nor multi-stakeholder scenarios, which is a feature that characterizes the solution presented in this paper.

In summary, while substantial work has been conducted to design publish-subscribe platforms in distributed systems, and to devise monitoring solutions specific for Beyond

5G systems, the key novelty of our approach is to bring the publish-subscribe paradigm into a Beyond 5G monitoring platform and to implement and evaluate experimentally the performance of the platform devised.

3 Monitoring architecture overview

A 5G scenario based on multiple sites, with heterogeneous components generating useful data that is likely to be monitored, relies on a flexible and distributed monitoring service in charge of collecting that monitoring data and distributing it to specific entities that obtain insights about the behavior of these components. In this sense, a general-purpose monitoring architecture is desired, so that it can fit in this kind of multi-stakeholder environments.

To start with the definition of this monitoring architecture, the main characteristics to be envisioned by the monitoring service have been extracted from a thorough analysis of the 5G EVE infrastructure and service requirements [8]. These are the following:

1. The monitoring distribution architecture must support multi-site network deployments involving distant sites.
2. The platform must deal with use cases that may generate monitoring data in the order of gigabytes.
3. Monitoring data have to be available to the verticals after the execution of the use case has concluded, estimating a retention time of at least 2 weeks.
4. Redundancy is needed to offer a fault-tolerant system.
5. The architecture must be flexible enough to accommodate a wide variety of elements to be monitored.
6. The support of some pre-processing techniques (e.g., translation across formats) may be needed for an efficient subsequent processing.
7. The collected metrics may be used and post-processed by a KPI Validation Framework,[1] which can also distribute the calculated KPIs' values from a specific set of metrics using this platform.

These features result in the architecture for the collection, distribution and pre-processing of monitoring data presented in Fig. 1, which satisfies all the requirements described above.

In this general-purpose architecture, two sets of components can be distinguished:

- In dark blue, some elements of the ***infrastructure*** to be monitored, included here for the sake of completeness, and which may be user equipment devices (UEs) (4G or 5G) radio antennas, physical network functions (PNFs), virtual network functions (VNF) or other components such as external monitoring tools which may provide monitoring data to the monitoring platform.
- In light blue, the elements that compose the monitoring platform itself, which will be presented next by following a bottom-up/west-east approach.

[1] This framework is out of the scope of this paper.

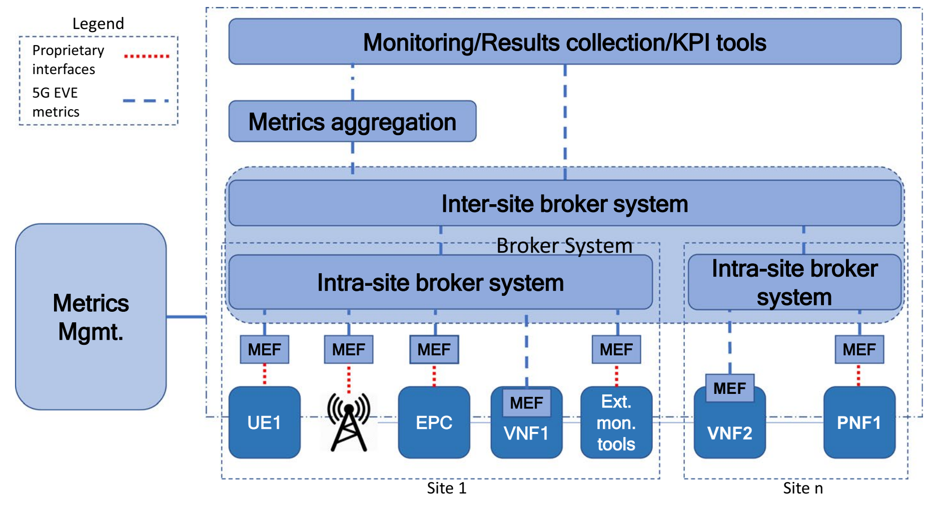

Fig. 1 Monitoring metrics architecture

The first component of the architecture to be described is the ***Metrics Management*** entity, whose main role is to properly configure the other components of the architecture, providing the configuration of the necessary data service function chains in order to enable metrics to be gathered, filtered, normalized and relayed to upper layers in the architecture to be further processed.

The component of the architecture directly connected to each element of the infrastructure is the ***Metrics Extractor Function (MEF)***, which takes care of extracting and translating (if required) the metrics generated by a heterogeneous set of infrastructure components. This module should be flexible enough to be integrated in different environments, from on-premises deployments to more agile facilities such as Edge environments.

Consequently, in order to be able to provide monitoring data to the monitoring system, a *MEF* must be integrated within the infrastructure component. In the proposed architecture, it is assumed that there is a one-to-one logical relationship between a particular *MEF* and its monitored infrastructure component, although this may be implemented in different ways, mainly depending if it is fully, partially or not integrated in the monitored components, as presented in Fig. 1. This is aligned with the distributed nature of the monitoring system; considering that, if a new element from the infrastructure of the 5G site needs to be monitored, it is just required the provision of a *MEF*. Moreover, as MEFs are intended to be lightweight software modules, they can be easily deployed and distributed along the infrastructure.

This modular design allows to have a dedicated *MEFs* per infrastructure device, which satisfies the requirement (5) explained before. This way, it would be possible to implement dedicated *MEFs* to handle any kind of proprietary interfaces (dotted red lines in Fig. 1). Then, the *Metrics Management* entity instructs each *MEF* to extract metrics from its monitored component and to make them available to the upper layer (i.e., the *Broker system*, which will be described next).

It is important to remark that all these metrics have to follow an homogeneous format (e.g., the one defined in [19]) to satisfy constraint (6) presented before. This might require a translation from a proprietary or different standard formats to the one used by the monitoring platform, in order to handle all the messages received from the *MEFs* in an unified way.

The monitoring data are then received by the ***Broker system***, which is in charge of storing and distributing not only the metrics obtained from different sites, but also the KPIs' values generated in upper layers. For accomplishing requirement (1), two brokering levels have been defined:

- The ***Intra-site broker***, deployed per site, whose role is to eventually harmonize the metrics' format to provide data in an unified way, preserving the data privacy of each site. By doing this, the distribution and extension of the architecture is achieved: as long as a 5G site is able to deploy a new *Intra-site broker*, collecting the monitoring data and providing it to upper layers, that 5G site could be then monitored.
- The ***Inter-site broker***, which is the central component of the *Broker system*, aggregating all the monitoring data received from each *Intra-site broker* to:
    - Aggregate metrics through the ***Metrics aggregation*** component, generating new metrics automatically based on those provided by the *MEFs*. For example, a given function may receive the instantaneous transmission rate at a given network interface every second, to then compute the mean rate in a 10-s window. More complex functions may estimate the average rate between two points in a defined window time.
    - Directly provide them to the different tools grouped in the ***Monitoring/Results collection/KPI tools*** entity, which is the entity consuming metrics from the *Metrics aggregation* or the *Inter-site broker system*, laying the ground for a set of value-added additional components that range from the KPI Framework for performance diagnosis already mentioned, which allows to fulfill requirement (7), to more complex modules such as data analytics platforms, SLA enforcement mechanisms or data visualization services, which can be fed from the monitoring data provided by the system.

Although the *Inter-site system* is a centralized entity in the architecture, this does not mean that the whole architecture is centralized itself. In fact, there can be as many *Intra-site broker* entities as 5G monitored sites, and there can also be as many *MEFs* as monitored elements within a site, being both of them the distributed units within the monitoring platform, as explained before.

Finally, in order to satisfy requirements (2), (3) and (4), the *Metrics Management* entity is the responsible for properly configuring all levels of the broker system in a per-deployment basis, also enabling the necessary security mechanisms to ensure that only the actors belonging to a given network deployment can manage the monitored data of their deployment and not others.

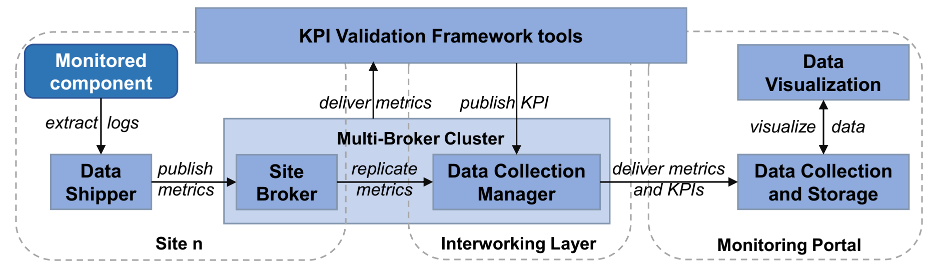

Fig. 2 Component chain that implements the general monitoring metrics architecture

4 Implementation based on the publish-subscribe paradigm

4.1 Detailed platform

Taking into account the general-purpose monitoring architecture described in Sect. 3, this section proposes a specific implementation of this design. The cornerstone of this platform is the publish-subscribe messaging pattern, providing a distributed system with parallel data processing capabilities which allows to meet the requirements imposed to the monitoring platform. As a result, this implementation results in the composition of a specific component chain, depicted in Fig. 2.

Thanks to the integration of the publish-subscribe messaging pattern, a multipoint-to-multipoint monitoring data flow is enabled, which is closer to a big data pipeline rather than to a classic relational database model, as a massive volume of data is pushed from site facilities without a specific format, which is not suitable to be stored in a relational model [20].

Following the above, the *Broker system* is mapped into a set of publish-subscribe queues, starting from local queues deployed in each site facility (*Intra-site broker*) that aggregate metrics to the Interworking publish-subscribe queue (*Inter-site broker*), which provides a transparent and seamless access to metrics' and KPIs' values from all sites to components from upper layers. In Fig. 2, each *Intra-site broker* is represented by a ***Site Broker*** entity, and the *Inter-site broker*, together with the *Metrics Management* service, is implemented by the ***Data Collection Manager*** component in the implemented architecture.

All the *Site Brokers* and the *Data Collection Manager* are based on *Apache Kafka* [21], an open-source, industry-proven publish-subscribe tool that manages data pipes and forwards the published data to the different components subscribed, providing a higher maximum sustainable throughput than other broker-based message-oriented middleware technologies [22]. Moreover, it also implements several useful functionalities related to data transformation and normalization (*Kafka Streams*), security (*Kafka ACL*) or data persistence (*Kafka Store*), among others [23]. This makes *Kafka* an optimal solution for data-movement, frequently adopted as pipe to different processing systems [24].

This hierarchical architecture can be encompassed with the so-called ***Multi-Broker Cluster***. In this way, the *Site Brokers* located in each site replicate the data received toward the *Data Collection Manager*, which is in charge of providing the data that come from different sources to the entities interested in consuming that data. This feature allows to deploy small processes in the sites that only gather monitoring data and forward it to upper layers of the platform, being then aligned with Edge's philosophy. In fact, this kind of publish-subscribe architecture has been already used in different approaches to transport messages in Edge platforms, as commented in Sect. 2.

The information model that defines the different topics that are handled by the *Multi-Broker Cluster* in a concurrent way is described in the so-called Topic framework proposal [23]. In that way, each topic is designed to manage a specific set of data (mainly related to a single metric or KPI to be monitored) that will be different to the data consumed by the other topics, enabling dataset isolation.

There are two main types of topics defined in the Topic framework, which are:

- Data topics, where each of them transports the values of the metric or KPI they refer to, followed by some meta information that may be useful for other modules. In particular, this information corresponds to the homogeneous format mentioned in Sect. 3, which specifies the fields that the message containing the data related to the metrics' and KPIs' values to be handled by the Monitoring platform must have.
- Signaling topics, used to deliver the name of data topics related to each metric or KPI to be monitored for a given network deployment. This is a function that fits in the scope of the *Metrics Management* service, which automates the process of creation and deletion of topics.

The components that interact with the *Multi-Broker Cluster* can be classified as publishers and subscribers, depending on whether they produce data to the publish-subscribe platform or they consume it. This distinction allows to simplify the workflow during the execution of a given use case, as subscribers only need to be subscribed to the topics related to the metrics and KPIs they want to consume data from (i.e., the ones used in the use case), and then, when a publisher produces data to these topics, the information is automatically delivered to the subscribers that are listening to the topics.

The main component which performs the metrics' data publishing operation is the ***Data Shipper***, playing the role of the *MEF* component from the general architecture, and whose objective is to execute the log-to-metric operation that transforms the heterogeneous, raw logs obtained from components and collection tools into metrics with a common, homogeneous format. These data shippers can be placed within each component as a lightweight software (ranging from general-purpose solutions already developed and packaged like *Beats* [25] to more complex solutions programmed for specific-purpose cases) or can be deployed in a separated server, but in both cases, they must be connected to the *Multi-Broker Cluster* with a logical connection. Again, this flexibility allows the *Data Shippers* to be deployed in a wide variety of environments, from Edge to Cloud.

Moreover, the ***KPI Validation Framework tools*** also contain publishers providing KPIs related to a given set of metrics received from the *Multi-Broker Cluster* after being published by specific *Data Shippers*, which means that these KPI tools also implement a subscriber for each metric to be consumed.

Finally, the ***Data Collection and Storage-Data Visualization*** component performs the expected functionalities provided by the *Monitoring/Results collection*

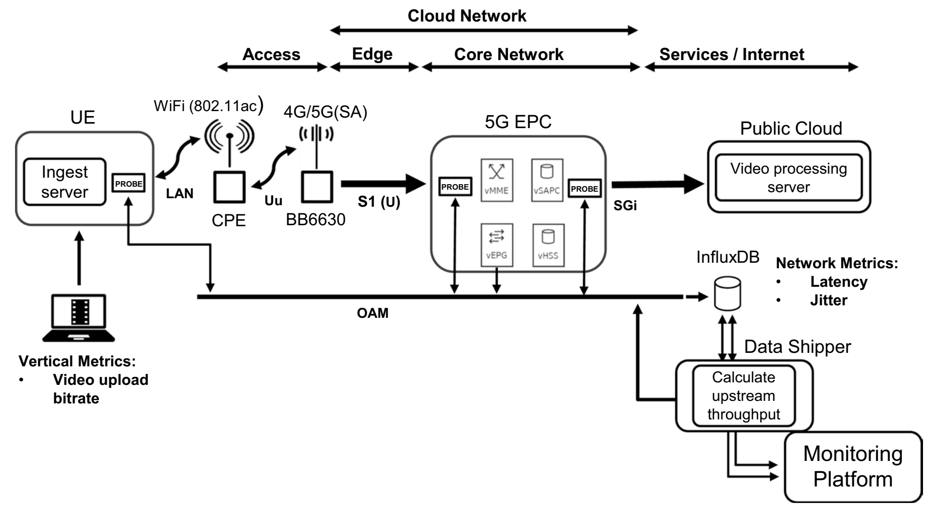

Fig. 3 Monitoring of the upstream throughput in a 5G use case. Adapted from [31]

entities with a solution based on the *Elastic (ELK) Stack* [26, 27]. This component receives the metrics' and KPIs' values through a specific subscriber for each metric and KPI, and it is separated logically in two main blocks [28, 29]:

- The ***Data Collection and Storage*** component, which collects each of the subscribed components metrics and KPIs, through *Logstash*, from the *Elastic Stack*, and provides data persistence, searching and filtering capabilities (related to the *Metrics aggregation* functionality from the general architecture) for obtaining the useful data to be monitored during the operation of the system thanks to *Elasticsearch*, also from the *Elastic Stack*.
- The ***Data Visualization*** component in charge of enabling the monitoring of the progress of the deployment in terms of that monitoring data displayed through *Kibana* from the *Elastic Stack*. For this purpose, a set of dashboards are created for each deployment, presenting the graphs related to each metric or KPI monitored.

4.2 Example of a monitored 5G service

To complete the description of the implementation of the monitoring platform, an example of a given 5G use case monitored by this system is presented. The full description of this 5G pilot, executed in the 5TONIC laboratory [30], is reported in [31], but the main aspects related to the execution of one of its use cases will be presented below for the sake of completeness.

In particular, this pilot refers to experiential tourism through 360-degree video and virtual reality, which uses both technologies to transform the experience of participants in events, conventions, presentations or meetings, amplifying their participation and improving their interactions.

The high-level implementation of the use case related to this pilot, which involves the upstream video distribution from user equipments to a service deployed in a public Cloud, using a 4G/5G network to distribute the traffic, is presented in Fig. 3.

In this use case, a high-quality video feed is ingested locally in the UE, which connects to the CPE via Wi-Fi. The CPE is 4G/5G capable and can connect to LTE or 5G SA RAN, both based on Ericsson technology. Once the connection is established, data are sent through radio to the 5G EPC, which forwards the traffic to the public Cloud in which the video processing server is located.

Moreover, a set of probes implemented by Ericsson in 5TONIC laboratory are deployed in key points of the network, in order to measure different network metrics such as the latency or the jitter. Then, a *Data Shipper* gets these metrics and calculates the corresponding upstream throughput values, which are sent to the monitoring system. This demonstrates the flexibility of the monitoring platform to monitor network metrics that can be captures from different points of the network architecture.

In preliminary tests, the results reported an uplink throughput of the 5G network with peaks near 60 Mbps, verifying a good performance and user perception of the video when demanding a video upload bitrate of 25 Mbps.

5 Performance evaluation

To assess and validate the proposed monitoring framework implementation, the testing process described below has been followed, based on the application of a top-down approach. In particular, it is based on the execution of specific experiments monitored by the monitoring platform, where an experiment is an emulation of a given network deployment, characterized by parameters like the bandwidth consumed.

As a result, the purpose of the performance evaluation process presented is to characterize the monitoring platform itself in terms of several performance parameters, which are related to the resource consumption of the components from the monitoring platform, and the latency and packet loss experienced by the experiments deployed, obtaining these two last parameters from a given throughput in the system. These performance parameters are well explained in Sect. 5.2.

So, the idea of this performance evaluation process is not related to present how the monitoring platform is capable of monitoring a given 5G service and show the results obtained for each monitored metric (which is already presented in Sect. 4, but it is a load testing process to check whether the designed and implemented monitoring platform is capable of handling a given amount of monitoring data (in terms of throughput and number of topics running in the platform, i.e., a certain number of metrics managed by the monitoring platform simultaneously), emulating real 5G use cases with a synthetic data rate and a specific number of topics running in the platform. This would help to eventually size the monitoring platform (in terms of number of servers, their hardware requirements, throughput to be handled, etc.) for a given deployment in 5G or Beyond 5G scenarios.

In this way, this performance evaluation process starts with **single-broker experiments** to characterize the platform in terms of several performance parameters and finishes with **multi-broker experiments** to check the impact of having the two brokering levels described in Sect. 3.

It has to be mentioned that, although the results are component-sensitive, because specific components with specific requirements and specific values of design parameters have been used, the procedure followed to do the test is not component-sensitive, but it is a general-purpose methodology that can be applied to other type of components.

5.1 System assumptions

The definition of the system under test (SUT) parameters is related to the 5G EVE multi-site platform's operation, where a set of network deployments derived from the different use cases defined in the project may be running simultaneously at a specific time, sharing all the computing and network resources provided by both the 5G EVE platform and the site facilities.

As a first approach to the evaluation, the following assumptions were made:

- The monitoring platform must be prepared to deal with extreme conditions, such as the simultaneous execution of a considerable amount of use cases. As the 5G EVE project initially proposes the validation of six specific use cases [8], the execution of a deployment from each use case at the same time can be taken as the worst case study to validate, resulting in six simultaneous deployments (i.e., experiments) to be handled by the monitoring platform.
- Each experiment can define a different number of metrics and KPIs to be collected and monitored during the execution of the use case, depending on vertical's needs. For this evaluation process, as these metrics can be extracted from different sources (e.g., UEs, VNFs, PNFs), and each source may have several related metrics or KPIs, and it can be assumed that each experiment will require the monitoring of an average of 20 parameters. Furthermore, as each monitored parameter has a topic assigned for the transport and delivery of their corresponding collected data, each experiment on average will create 20 topics in the monitoring platform. As a result, the maximum number of topics[2] created in the platform would be $20 \times 6 = 120$ in this case.
- The size and the publication rate of the messages containing the values of metric or KPI managed by the monitoring platform depend on the nature of the data transported. As a result, four different alternatives have been considered for the tests:
  - 100 B and 1 KB messages for data traffic (i.e., numeric or string values), representing the 80% of all the monitoring traffic (40% for each case). The publication rate for both options is set to 1000 $messages/s$.
  - 100 KB and 1 MB messages for multimedia traffic (i.e., images or video frames), which would be the remaining 20% (10% for each case). The publication rate for both cases is less than the data traffic one, as the received throughput almost never reached that value due to the message size, with 10 $messages/s$ for 100 KB messages and 1 $message/s$ for 1 MB messages.

[2] This figure does not include the signaling topics presented in Sect. 4, whose footprint is not significant compared to these data topics.

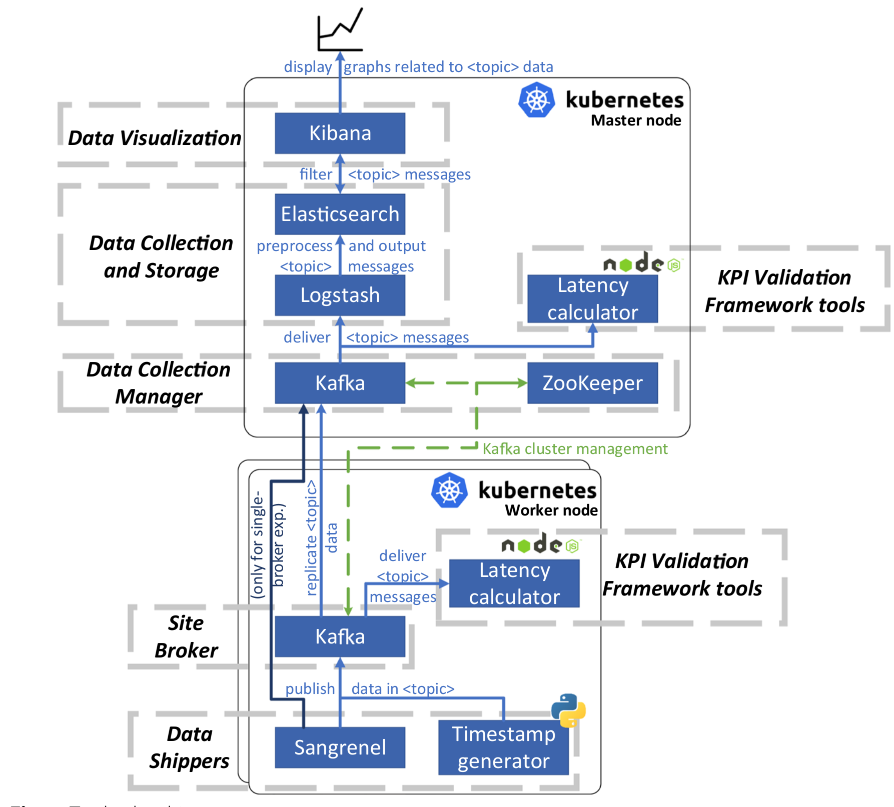

Fig. 4 Testbed architecture

The percentages have been selected assuming that most of the data will be small-side messages, but also considering that there may be larger messages, mainly related to multimedia data. As a result of the figures selected for each kind of message, this results in a concurrent publication rate of approximately 102,4 Mbps per experiment.

- Another important parameter related to the publishers is the message batch size, which controls the amount of messages to collect before sending messages to the *Multi-Broker Cluster*, and which was set to 1 after validating that higher values of this parameter cause worse results in terms of latency.
- The selected values of publication rate for each message size are also coherent for the subsequent calculation of the disk size estimation for each broker node, which is computed as $D = s \times r \times t \times f/b$, where s is the message size, r is the publication rate, t is the retention time (at least 2 weeks, as discussed in Sect. 3), and f and b are both the replication factor and the number of brokers in the system, typically $f = b - 1$, this leading to a value slightly below 100 TB, which is an estimation of the expected amount of data handled in the project.

5.2 Testbed setup

The testbed used for the evaluation of the architecture consists on a set of *Ubuntu Server 16.04* virtual machines (VMs) [32] deployed in a server located in the 5G EVE Spanish site facility, 5TONIC, using *Proxmox* [33] as virtualization environment, and *K3s* (a lightweight *Kubernetes* distribution) [34] to orchestrate the containerized components[3] deployed in each VM. This server is equipped with 40 Intel(R) Xeon(R) CPU E5-2630 v4 at 2.20 GHz and 128 GB RAM. The distribution of components in each VM can be seen in Fig. 4.

The proposed scenario intends to simulate the monitoring and data collection process of the metrics and KPIs related to a set of network deployments. The components deployed in each VM are the following:

- *Kubernetes* Worker node VMs: each *Kubernetes* worker emulates a site, including *Data Shippers* for publishing monitoring data in a *Site Broker*, based on *Apache Kafka*, that replicates the data toward the *Data Collection Manager*, placed in the *Kubernetes* Master node. Regarding the *Data Shippers*, this role is played by two components:

  - *Sangrenel* [36], which is a *Kafka* cluster load testing tool that allows to configure parameters such as the message/batch sizing and other settings, writing messages to a specific topic and obtaining, as output, the input message rate (used for calculating the **input/output (I/O) message rate**, i.e., the received throughput divided by the publication rate) or the **batch write latency** (i.e., time spent until receiving an ACK message from the broker), which are some of the performance parameters under study, being dumped every second.
  - A *Python*-based *Timestamp generator* [37] used exclusively in multi-broker experiments. It sends messages with timestamps embedded that are eventually received by a *Latency calculator* component, based on *Node.js*[4] [39], which takes the timestamps and calculates the so-called **broker latency**, i.e., time spent between the publication of data and its reception in an entity subscribed to the *Site Broker*. In fact, this component can be associated with the *KPI Validation Framework tools*, as it calculates the latency (KPI) based on timestamps (metric).

- *Kubernetes* Master node VM: in this server, the *Data Collection Manager*, *Data Collection and Storage* and *Data Visualization* components from Fig. 2 have been implemented with a solution based on *Apache Kafka* and the *Elastic Stack*. A *ZooKeeper* [40] instance is also running to coordinate the *Kafka* cluster, and there is also another instance of the *Latency calculator* deployed here to calculate the **end-to-end latency** KPI, this being the time spent between the publication of data in a given site and its reception in an entity subscribed to the *Data Collection Manager* (so that data have been previously replicated from the *Site Broker*).

[3] The images of these components can be found in [35].

[4] This programming language has been used in order to make use of *Kafka*'s KIP-392 feature, to receive data from the closest replica [38].

For monitoring the **resource consumption** of each container (focusing on the **CPU consumption**), *Docker* [41] native tools (e.g., *docker stats*) have been used.

5.3 Single-broker experiments

For these experiments, only one *Kafka* broker is required, so the testbed depicted in Fig. 4 can be simplified by only using one *Kubernetes* Worker node with just a *Sangrenel* container directly connected to that *Kafka* broker represented with the dark blue line that connects both components in the testbed diagram.

5.3.1 Experiments with one topic

To start with the performance analysis of the monitoring platform, experiments with only one topic created were performed, checking that the system operates correctly and consistently for each message size and publication rate proposed in Sect. 5.1 without limit of resources, and also with the objective of defining the minimum set of computing resources (RAM and vCPU) for the most critical components of the architecture.

In this set of tests, some of the assumptions from the system characterization were confirmed, e.g., the poor results for multimedia traffic when its publication rate is 1000 *messages*/*s*, where the I/O message rate falls from 1 (obtained when the reduced publication rate is used) to 1/4 in the best-case scenario, or that the optimal value for the message batch size parameter is 1 for all types of traffic, as increases in their order of magnitude cause exactly the same increase in the order of magnitude of latency. For example, for a 100 B message size, the batch write latency goes from 0.8 ms with a message batch size of 1–500 ms, where the message batch size is 1000.

Apart from that, it was also observed that the resource consumption in the components of the monitoring architecture is CPU intensive for the most critical components of the platform, which are *Kafka*, *Logstash* and *Elasticsearch*, leaving the RAM to work as buffer and cache before persisting data to disk. As a consequence, this fact facilitates the sizing of these components, as the RAM value can be fixed with a specific value (in this case, with 2 GB of RAM is enough for working properly during the testing process), whereas the CPU value is the only variable term.

In terms of CPU, for a single-topic experiment, *Logstash* is the most critical component, with a consumption that ranges from 100 to 200%, requiring 4 vCPU in order not to degrade the performance. However, the CPU consumption in *Kafka* and *Elasticsearch* stays below 100% for all types of traffic, so 1 vCPU for both of them should be enough to cover single-topic experiments. However, in multi-topic experiments, which will be studied next, *Kafka* becomes the most critical component with a noticeable increase in its CPU consumption, whereas *Logstash* and *Elasticsearch* approximately maintain the same consumption profile.

5.3.2 Experiments with multiple topics

In multi-topic experiments, the distribution of performance parameter values between topics of the same type (i.e., that handle the same type of data, message size and publication rate) in a given experiment is expected to be uniform in general conditions, where there are no more priority topics than others.

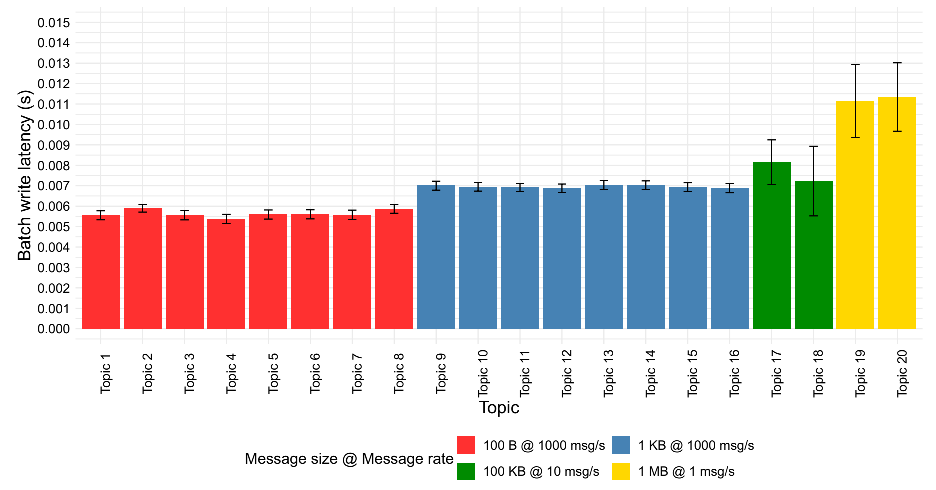

Fig. 5 Batch write latency distribution in a network deployment with 20 topics

This assumption is confirmed in Fig. 5 for the batch write latency analysis in one experiment with multiple topics, according to the per-network deployment topic distribution described in Sect. 5.1. As a result, this confirmed assumption is used in subsequent tests for accumulating and averaging the values obtained from performance parameters in topics of the same type, as if they were a single topic, which allows to simplify the performance analysis. Moreover, in Fig. 5, it can be also observed that latency is higher in larger message traffic, also increasing the deviation of the results, as seen in the longer 95% confidence interval estimated for multimedia traffic, for example. This reflects that smaller messages result in better and more precise values of latency.

Continuing with the different tests carried out related to multi-topic experiments, they aim at evaluating two design parameters that causes variations in the monitoring platform's workload: (1) the number of topics created and running in the system as concurrent processes, due to the execution of simultaneous deployments, and (2) the total throughput received by the monitoring system, calculated as the sum of all input message rates received from each topic.

However, a variation in any of these design parameters may cause different effects in the system in terms of CPU consumption or performance that must be characterized, also checking whether the superposition property can be applied when both parameters are modified simultaneously. For doing this, the study was divided into two parts:

1. A first analysis where one of the design parameters is modified, while the other one stays fixed.
2. A final test including the modification of both parameters at the same time, checking whether the superposition of individual effects is present.

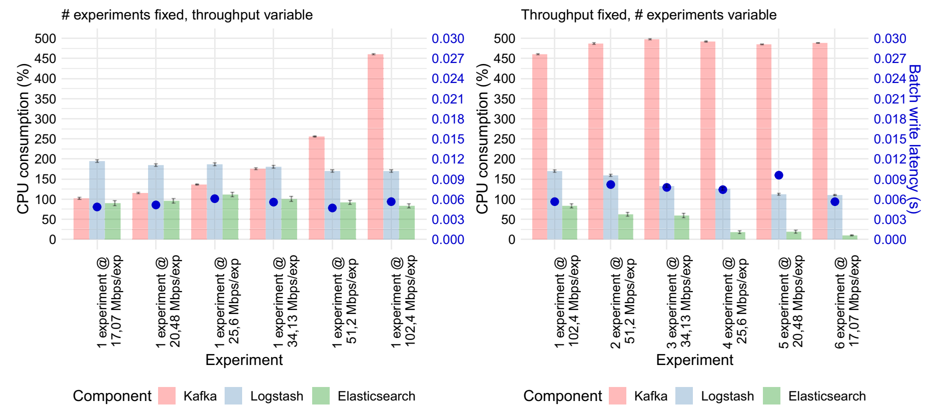

Fig. 6 CPU consumption and batch write latency evolution for 100 B data traffic in different experiments

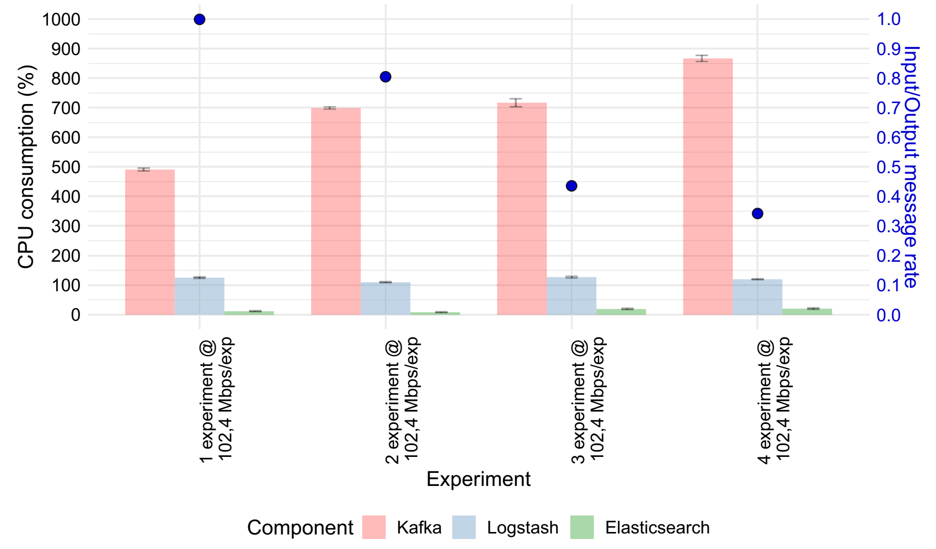

Fig. 7 CPU consumption and I/O message rate evolution for 100 B data traffic in different experiments

Part (1) is presented in Fig. 6, where the CPU consumption and the batch write latency related to 100 B aggregated data traffic[5] are evaluated for different examples of experiments:

- On the left side, the number of experiment is fixed in 1, whereas the total throughput is modified, using the theoretical input message rate as upper limit (i.e., 102,4 Mbps) and dividing it by values between 1 and 6.
- On the right side, the number of experiments is variable, ranging from 1 to 6, but the total throughput for all deployments is conserved, which is achieved by dividing the message rate aforementioned by the number of experiments deployed.

[5] This size is used in the rest of the analysis because it presents a lower value of latency with a tighter 95% confidence interval, according to Fig. 5.

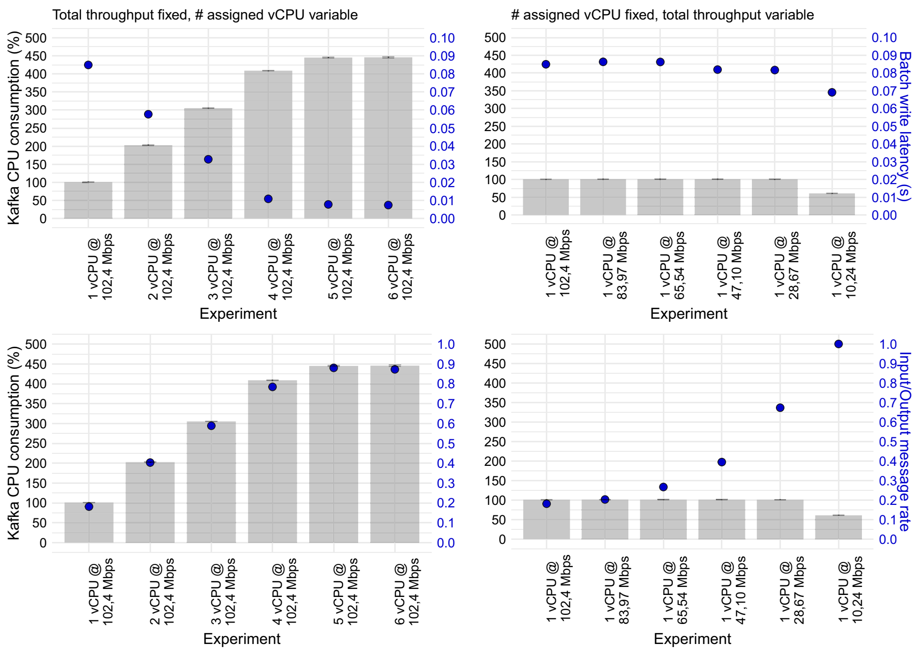

Fig. 8 Effect of saturation in performance parameters when limiting *Kafka* vCPU allocated in different experiments

In both cases, it is observed that the batch write latency does not vary when modifying one of the design parameters, and it is also true for the I/O message rate, which tends to 1. However, in the first case, when the total throughput becomes higher, the *Kafka* CPU consumption increases with a trend that seems exponential, but in the second case, the CPU consumption also remains constant in average.

As a result, while the total throughput has an effect in the *Kafka* CPU consumption with an exponential tendency, the number of network deployments (i.e., the number of topics in the system) does not seem to influence the system performance, as long as the total throughput is conserved when there is an increase in the number of topics, taking care of specifying correctly the publication rate in order not to exceed the system limits. However, this is true while the system is not saturated. When this happens, the effect is similar to the one shown in Fig. 7, related to the part (2) of the study aforementioned.

Here, when the number of network deployments increases, the total throughput is also higher, and in fact, it can be noticed that message loss is present from two experiments deployed, as the I/O message rate is nearly 0,8 (so 20% of the messages are lost), and falling until less than 0,4 in the case of four experiments deployed simultaneously, value that remains constant even if more experiments are deployed (these experiments have not been included in Fig. 7 just to present the saturation process with more detail).

The evolution of the CPU consumption in *Kafka* is also stopped due to this saturation state, as well as the latency starts to present variations as it is calculated based on the messages that are eventually received.

In fact, these results are quite aligned with the outcomes from [42], where it was reported that *Kafka* throughput depends linearly on the number of topics, reaching a hard limit at some specific point. According to this study, when there is only one *Kafka* replica, the limit is reached for around 15.000–20.000 *messages*/*s*, value which is close to these results, as one experiment in our testbed means around 16.000 *messages*/*s* and a second deployment causes a loss of performance, since that limit, which should be between 16.000 and 32.000 messages per second, is exceeded.

This issue related to effects caused by resources' saturation must be also taken into account in order to introduce these CPU-bound components in Edge environments, where the number of physical and virtual resources allocated to execute these workloads are quite limited. In this way, apart from having a theoretical limit imposed by the technology itself, the amount of resources can also have an impact on performance in case of sizing the platform wrongly, provoking a loss of performance even before reaching the hard limit.

To reflect the impact on performance caused by the limitation on computing resources (i.e., vCPU allocation in the *Kafka* container), Fig. 8 presents the evaluation of both the batch write latency (top subplots) and the I/O message rate (bottom subplots), for 100 B data traffic, in two situations:

- First of all, assuming that a full experiment is being executed in the platform (i.e., a total throughput of 102,4 Mbps is received by *Kafka*), the vCPUs assigned to the *Kafka* container was modified from 1 to 6 (the two graphs on the left in Fig. 8); checking that, from 5 vCPU, the values obtained for the performance parameters become reasonably good and stable.
- However, on a scenario where the *Site Broker* is placed in the Edge, a high-resource allocation cannot be guaranteed. For this reason, a new set of tests in which the vCPU allocation was fixed to 1 vCPU, then varying the throughput received by *Kafka*, was carried out (the two graphs on the right in Fig. 8). The values used for the throughput vary between the 100% and the 10% of the throughput related to an experiment (i.e., 102,4 Mbps). The results reflect that, although the latency does not improve when a lower throughput is received, this is not the case for the I/O message rate, which improves every time that throughput is reduced until reaching a value of 1 when the throughput is reduced to the 10%.

Consequently, to move to an Edge environment, it is crucial to limit the resource allocation, but also the throughput received by the monitoring platform, in order to avoid packet loss. This issue should not be a problem in Edge environments, assuming that most use cases deployed in this kind of scenarios will prioritize the ability to support a large number of connections rather than guaranteeing a certain value of latency or bandwidth; as happens in IoT, for example. Therefore, the higher values of latency, compared to the ideal scenario in which there are no problems related to resource consumption (70 ms vs. 10 ms, approximately), should not be a problem, while the throughput is kept at a reasonable value. In this case, this limit can be set to 10 Mbps.

5.3.3 *System scalability validation*

To avoid the saturation effect presented in Figs. 7 and 8, the direct solution is to build mechanisms and processes that allow system scalability, mainly oriented to the

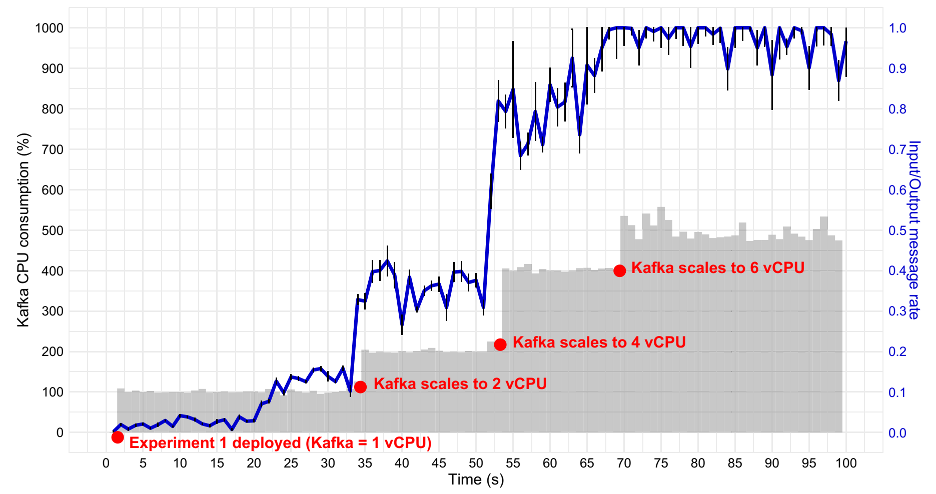

Fig. 9 I/O message rate evolution in an experiment with vertical scaling mechanisms enabled

application of horizontal and/or vertical scaling processes depending on the current status of the platform.

For this evaluation process, a preliminary vertical scaling system for the central component of this monitoring platform is proposed (i.e., no new instances are added, but the computing resources attached to the available instance are increased or decreased depending on the workload), based on the results obtained in the previous tests as training data, used to refine the different cases that can occur in terms of resource consumption (mainly related to CPU) and performance evaluation (mainly based on the batch write latency and the I/O message rate), and the conditions related to each case that trigger the system scale process.

Figure 9 presents an example of vertical scaling for one experiment deployed in the platform. In this case, the *Kafka* container is scaled by increasing its vCPU assignment until the system is able to handle the workload received without saturating, decision that depends on different parameters, such as, e.g., the current CPU consumption, the delay to compute a KPI or some other performance variable.

Note that, in this case, for illustrative purposes, an upscale is only triggered when a CPU is fully occupied for relatively long periods of times, this resulting in a relatively high convergence time (around one minute) of the I/O message rate, but more "agile" schemes could be easily implemented if needed.

5.4 Multi-broker experiments

Finally, the scalability of the full distributed, multi-site platform, as built in the testbed already presented in Fig. 4, will be evaluated in terms of the performance parameters already presented in Sect. 5.2 and the CPU consumption of the *Data Collection Manager*'s *Kafka* broker, whose computing resources will not be limited. On the other hand, the *Site Brokers* will be limited to 1 vCPU, taking the value already tested in the tests presented in Fig. 8.

In this case, the meaning of experiment will be a bit different. This way, each experiment instantiated in multi-broker experiments will be executed in a particular

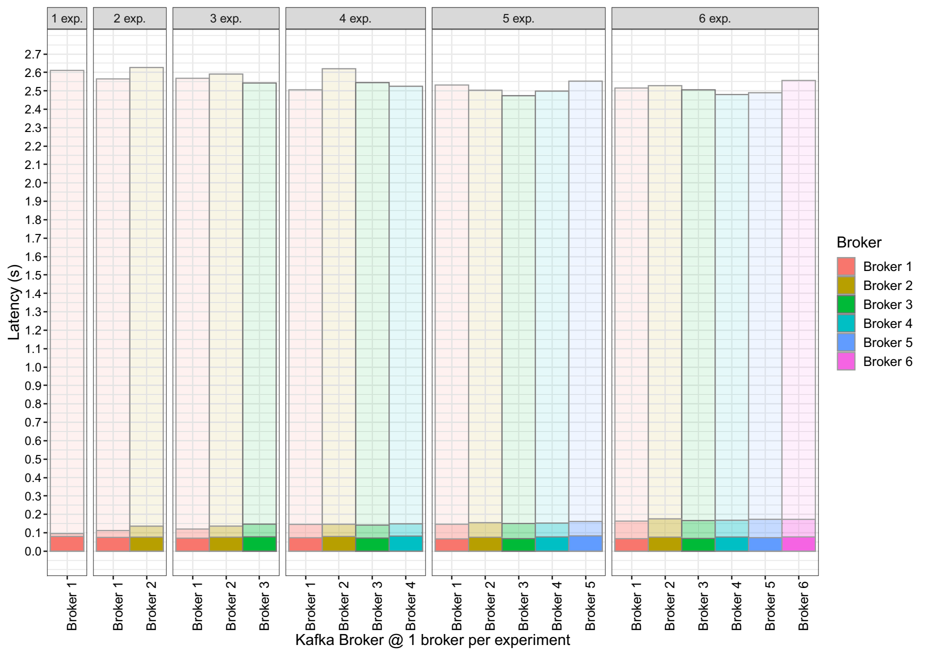

Fig. 10 Evolution of the three types of latency in multi-broker experiments

Kubernetes Worker node (so, for six experiments, six *Kubernetes* Worker nodes will be required), sending monitoring data to the corresponding *Site Broker* at 10% of the total throughput calculated in Sect. 5.1 (i.e., 10,24 Mbps), which is the throughput hard limit to avoid saturation, as stated in Fig. 8.

5.4.1 Impact on latency

The first performance parameter to be evaluated is the latency, in the different acceptations that are defined in Sect. 5.2: the batch write latency, the broker latency and the end-to-end latency. The values obtained during the execution of experiments, from one to six, for 100 B data traffic, can be seen in Fig. 10. Here, a similar effect than the one obtained in Fig. 5 can be observed: the results obtained in each site are similar for each case, so that performance data can be also aggregated in future analysis.

Moreover, the same tendency in latency values than observed in Fig. 8 can be seen also here: the latency does not vary even though the total throughput received by the monitoring platform increases due to the creation of new experiments.

Furthermore, the results[6] obtained for each type of latency are consistent with the definition of each of them: it is expected that the batch write latency (the darker color for each case) would give the lowest value (approx. 70–80 ms), as it only implies the reception of the ACK from the *Site Broker*. The next one would be the broker latency (the color of "intermediate" darkness in the graph), in which the *Site Broker* has also to deliver the data to a subscriber, but it can be checked that this does not cause a

[6] Note that these results have been obtained in a virtualized scenario, in which the latency between virtual machines and containers is negligible. In a real scenario, the delay introduced by each of the path components must be also taken into account.

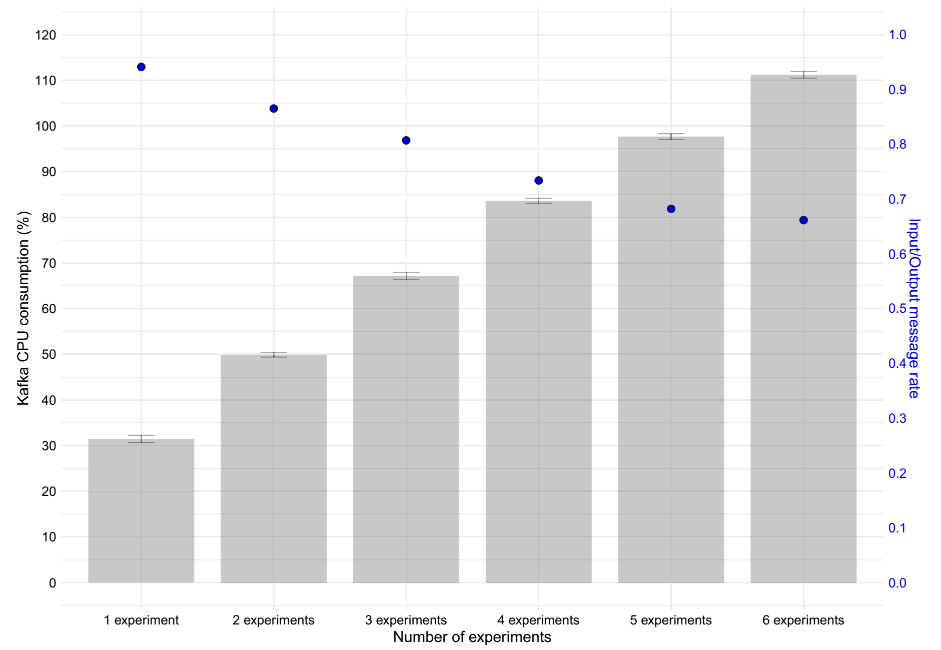

Fig. 11 Evolution of data collection manager CPU consumption and I/O message rate in multi-broker experiments

great impact on latency, as it is increased to nearly 120 ms in the worst case. And finally, the highest value on latency (approx. 2.5–2.6 s) is obtained for the end-to-end latency (the lighter color in the graph), due to the replication operation performed between each *Site Broker* and the *Data Collection Manager* and also the delivery to the corresponding subscriber. This value can be assumed in Edge environments for the reasons aforementioned.

5.4.2 *Impact on CPU consumption and packet loss ratio*

Finally, the impact on the I/O message rate in the multi-broker experiments is the same than experienced in single-broker experiments with CPU limitation (reflected in Fig. 8), where the packet loss increases with the increase in the total throughput received in the platform. This effect can be seen in Fig. 11, where the performance results from different brokers have been aggregated due to the results obtained in Sect. 5.4.1.

It can be observed that I/O message rate falls to nearly 0.65 when the six experiments are being executed concurrently, meaning a total throughput received of around 60 Mbps. This result, compared to the case observed in Fig. 8 with a single broker, with 1 vCPU, consuming 65,54 Mbps (the I/O message rate was less than 0.3), implies that the distribution of the total throughput between several *Site Brokers* improves the results.

Moreover, the CPU consumption in *Data Collection Manager*'s *Kafka* broker also increases with each experiment, but in a less rate, reaching the 110% of vCPU consumption for six experiments. Consequently, although the core of the Monitoring platform is intended to be executed in environments without limit of computing resources, this final result may allow the deployment of some components of this core (e.g., the *Data*

Collection Manager) on the Edge; as long as the total throughput, again, does not exceed a specific limit that causes saturation (60 Mbps in this case).

6 Results and discussion

The performance evaluation process performed in Sect. 5 has revealed some interesting insights related to the monitoring architecture. The first one is that the distribution of the performance parameter values in topics of the same type is uniform in both single-broker and multi-broker configurations, allowing the aggregation of the performance values obtained for each topic of the same type and, as a result, simplifying the study of the overall system.

In single-broker experiments, it has been also detected that the total throughput is the parameter that can cause the greatest impact on system performance, with two different possibilities: while the system has enough free resources to work, the CPU consumption tends to increase exponentially, keeping batch write latency and I/O message rate constant. However, when the system is saturated, which seems to happen for a total throughput between 16.000 and 32.000 packets per second, this exponential growth is stopped and the I/O message rate fails below 0,4 in the worst case.

After detecting this, the analysis of the performance parameters when the computing resources allocated (i.e., the vCPU) are limited revealed that the system can reach the saturation state even before that the theoretical limit aforementioned. This constraint can be regulated with the modification of the total throughput injected in the platform, allowing to increase the I/O message rate by reducing the throughput, while maintaining lower resource's usage and a practically constant latency. This is particularly important in the transition toward more flexible deployment such as Edge-based environments, in which resource's consumption is a crucial issue to be tackled. Furthermore, these results were used to build a preliminary vertical scaling mechanism, which calculates how many resources are needed for a given workload.

Finally, in multi-broker experiments, the impact of instantiating several network deployments, consequently involving the joint activity of different *Kafka* brokers, was evaluated, checking that the latency, in its different variants, remains also constant, being then the I/O message rate the performance parameter to be optimized by adjusting again the total throughput received by the platform, issue that should be easy to solve in Edge environments, where latency and bandwidth are not as important as a flexible deployment of solutions to ensure a lower consumption, allowing the connectivity of a huge set of devices to a given platform.

7 Conclusions and future research

This paper has presented a modular monitoring architecture, flexible enough to easily accommodate to different network deployments. Moreover, an implementation based on the publish-subscribe paradigm has been also proposed, confirming that is able to manage real, complex network deployments in both single-broker and multi-broker configurations. Finally, based on the results obtained after this performance evaluation process, it has been confirmed that this monitoring platform would be able to scale in multi-site scenarios, enabling also lightweight deployments oriented to Edge and Beyond 5G deployments.

Despite this, a saturation effect due to a software limitation related to the technology used (i.e., *Kafka*) has been also presented in the performance evaluation process. This does not imply that the monitoring system becomes unrealistic; in fact, this saturation effect is a predictable behavior, as it has been already described in previous work from the state of the art [42]. This problem can be solved from different points of view; for example, by applying scaling techniques to better size the platform. As presented in Sect. 5.3.3, when using simple vertical scaling techniques (which does not make a great impact in terms of using additional resources), the performance of the system improves considerably, preventing the monitoring platform for losing performance due to this software limitation. Another example could be to change the technology used, but as reflected in the state of the art [22], Kafka is the best tool related to publish-subscribe mechanisms.

As a consequence of these results obtained during the evaluation process, several topics for future research can be defined. Some of them, which were declared in the work that is the base of this research [1], have already been analyzed and fulfilled; such as the execution of real network deployments in the monitoring platform [31, 43] or the enhanced implementation of the monitoring platform, which is available in the 5G EVE Github repository [44].

However, there are still some pending issues to be studied in the medium term that would enable the improvement of the platform in terms of performance; for example, the usage of artificial intelligence (AI) and machine learning (ML) techniques in order to improve the system scalability process, thus being able to allocate new compute resources based on the information extracted and analyzed from the network.

Another topic to have in mind is the alignment with standardization efforts; not only in the terms explained in [1], where some gaps detected in 3GPP standards were presented, trying to fit the monitoring platform in them, but also having in mind other initiatives, such as the ETSI-NFV platform for the management and orchestration of network functions deployed in a given infrastructure. In that case, the monitoring platform may help in the collection of metrics from different sources (infrastructure, VNFs, etc.) to easily deliver them to the entities interested in that data; for example, data analytics components, linking with the integration of AI and ML technologies aforementioned.

Finally, even though the system has been validated in a testbed that uses some technologies which are oriented to Edge environments, such as *K3s*, also extracting some useful conclusions related to these scenarios in Sect. 5, it is true that a real implementation that operates in Edge environments is still missed, but the components needed to perform that deployment have already been developed and are publicly available [35], so it would be just a matter of finding a proper use case that may need this functionality in order to perform and test the integration in a real case.

8 Methods/experimental

Although the performance evaluation process is fully detailed in Sect. 5, some guidelines related to this procedure will be summarized below.

In this research work, the monitoring platform presented in this study implements a solution based on the publish-subscribe paradigm, which is analyzed in different types of deployments, ranging from single-broker network deployments, in which only the first level of brokering of the architecture is tested, to multi-broker deployments, where the full clusterized solution is evaluated.

To do this, some performance metrics related to latency and packet loss are analyzed, together with the resource consumption of some of the components of the platform. These parameters can be extracted by using specialized tools and Linux commands during the execution of the use cases.

Each deployment (called *"experiment"* in terms of the performance evaluation process) involves the injection of a workload at a given data rate in the monitoring platform. The duration of the experiments can be controlled by using scripts designed for that purpose. In general terms, each experiment lasts 5 min, and the data obtained for each performance parameter can be analyzed by using statistical measures like the mean, variance and standard deviation.

Abbreviations
2G: Second generation; 3GPP: Third-generation partnership project; 4G: Fourth generation; 5G: Fifth generation; 5Gr-VoMS: 5Gr-vertical-oriented monitoring system; 6G: Sixth generation; ACK: Acknowledgement; ACL: Access control list; AI: Artificial intelligence; B: Byte; CPE: Customer premise equipment; CPU: Central processing unit; ELK: Elasticsearch Logstash Kibana; eMBB: Enhanced mobile broadband; ENI: Experiential networked intelligence; EPC: Evolved packet core; ETSI: European Telecommunications Standards Institute; GB: Gigabyte; GHz: Gigahertz; I/O: Input/output; IMT: International Mobile Telecommunications; IoT: Internet of Things; ISG: Industry Specification Groups; ITU-R: International Telecommunication Union-Radiocommunication; KB: Kilobyte; KPI: Key performance indicator; LTE: Long-term evolution; MB: Megabyte; Mbps: Megabits per second; MDAF: Management data analytics function; MEC: Multi-access edge computing; MEF: Metrics extractor function; ML: Machine learning; mMTC: Massive machine-type communications; ms: Millisecond; NFV: Network function virtualization; NWDAF: Network data analytics function; ONAP: Open network automation platform; O-RAN: Open radio access network; PNF: Physical network function; RAM: Random access memory; RAN: Radio access network; SA: Standalone; SDO: Standard development organization; SLA: Service-level agreement; SUT: System under test; TB: Terabyte; UE: User equipment; URLLC: Ultra-reliable and low latency communications; VM: Virtual machine; VNF: Virtual network function; ZSM: Zero touch network and service management.

Acknowledgements
Not applicable.
This paper is an extension of the following paper: [1].

Authors' contributions
RP was the main responsible for the design, implementation and evaluation of the monitoring platform presented in this paper, JGR contributed to the design and implementation of the monitoring architecture, AZ contributed to the design, implementation and evaluation of the system, PS contributed to the design of the experiments and the review of the related work, and AB contributed to the problem formulation and context. All authors read and approved the final manuscript.

Funding
This work was partly funded by the European Commission under the European Union's Horizon 2020 program—Grant Agreement Number 815074 (5G EVE project). This work was also partly funded by the Community of Madrid, under the grant approved in the ***"Convocatoria de 2017 de Ayudas para la Realización de Doctorados Industriales en la Comunidad de Madrid (Orden 3109/2017, de 29 de agosto)"***, Grant Agreement Number IND2017/TIC-7732. The paper solely reflects the views of the authors. Neither the European Commission nor the Community of Madrid are responsible for the contents of this paper or any use made thereof.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Telcaria Ideas S.L., C/ Barrionuevo, 8, 28911 Leganés, Madrid, Spain. [2] Department of Telematic Engineering, Universidad Carlos III de Madrid, Av. de la Universidad, 30, 28911 Leganés, Madrid, Spain. [3] IMDEA Networks Institute, Av. del Mar Mediterraneo, 22, 28918 Leganés, Madrid, Spain.

Received: 19 October 2020 Accepted: 24 March 2021
Published online: 07 April 2021

References

1. R. Perez, J. Garcia-Reinoso, A. Zabala, P. Serrano, A. Banchs, A monitoring framework for multi-site 5G platforms, in *2020 European Conference on Networks and Communications (EuCNC)* (2020), pp. 52–56
2. S.E. Elayoubi, M. Fallgren, P. Spapis, G. Zimmermann, D. Martín-Sacristán, C. Yang, S. Jeux, P. Agyapong, L. Campoy, Y. Qi, et al., 5G service requirements and operational use cases: analysis and METIS II vision, in *2016 European Conference on Networks and Communications (EuCNC)* (IEEE, 2016), pp. 158–162
3. X. Foukas, G. Patounas, A. Elmokashfi, M.K. Marina, Network slicing in 5G: survey and challenges. IEEE Commun. Mag. **55**(5), 94–100 (2017)
4. ITU-R: IMT Vision—framework and overall objectives of the future development of IMT for 2020 and beyond. Recommendation ITU-R M.2083-0 (2015). https://www.itu.int/rec/R-REC-M.2083-0-201509-I
5. A. de la Oliva, X. Li, X. Costa-Perez, C.J. Bernardos, P. Bertin, P. Iovanna, T. Deiss, J. Mangues, A. Mourad, C. Casetti, J.E. Gonzalez, A. Azcorra, 5G-TRANSFORMER: slicing and orchestrating transport networks for industry verticals. IEEE Commun. Mag. **56**(8), 78–84 (2018)
6. M. Gupta, R. Legouable, M.M. Rosello, M. Cecchi, J.R. Alonso, M. Lorenzo, E. Kosmatos, M.R. Boldi, G. Carrozzo, The 5G EVE end-to-end 5G facility for extensive trials, in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)* (IEEE, 2019), pp. 1–5
7. J. Garcia-Reinoso, M.M. Roselló, E. Kosmatos, G. Landi, G. Bernini, R. Legouable, L.M. Contreras, M. Lorenzo, K. Trichias, M. Gupta, The 5G EVE multi-site experimental architecture and experimentation workflow, *in 2019 IEEE 2nd 5G World Forum (5GWF)* (IEEE, 2019), pp. 335–340
8. 5G EVE: requirements definition and analysis from participant vertical industries. Deliverable D1.1 (2018). https://doi.org/10.5281/zenodo.3530391
9. 5G EVE: 5G EVE end to end reference architecture for vertical industries and core applications. Deliverable D1.3 (2019). https://doi.org/10.5281/zenodo.3628333
10. C. Papagianni, J. Mangues-Bafalluy, P. Bermudez, S. Barmpounakis, D. De Vleeschauwer, J. Brenes, E. Zeydan, C. Casetti, C. Guimarães, P. Murillo, A. Garcia-Saavedra, D. Corujo, T. Pepe, 5Growth: AI-driven 5G for automation in vertical industries, *in 2020 European Conference on Networks and Communications (EuCNC)* (2020), pp. 17–22
11. 3GPP: architecture enhancements for 5G system (5GS) to support network data analytics services (Release 16). TS 23.288 v16.1.0 (2019). https://www.3gpp.org/DynaReport/23288.htm
12. 3GPP: management and orchestration; architecture framework. TS 28.533, v16.2.0 (2019). https://www.3gpp.org/DynaReport/28533.htm
13. O.-R. Alliance, O-RAN: towards an open and smart RAN. White paper (2018)
14. ETSI: network transformation; (orchestration, network and service management framework). White paper no. 32 (2019)
15. 3GPP: study on integration of open network automation platform (ONAP) and 3GPP management for 5G networks (Release 16). TR 28.890 v16.0.0 (2019). https://www.3gpp.org/DynaReport/28890.htm
16. A. Javed, K. Heljanko, A. Buda, K. Främling, CEFIoT: a fault-tolerant IoT architecture for edge and cloud, in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)* (2018), pp. 813–818
17. C. Martín, D. Garrido, M. Díaz, B. Rubio, From the edge to the cloud: enabling reliable IoT applications, *in 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)* (2019), pp. 17–22
18. Q. Yuan, X. Ji, H. Tang, W. You, Toward latency-optimal placement and autoscaling of monitoring functions in MEC. IEEE Access **8**, 41649–41658 (2020)
19. 5G EVE: second implementation of the interworking reference model. Deliverable D3.4 (2020). https://doi.org/10.5281/zenodo.3946323
20. 5G EVE: interworking reference model. Deliverable D3.2 (2019). https://doi.org/10.5281/zenodo.3625689
21. Apache Kafka. https://kafka.apache.org/. Accessed 26 Feb 2021
22. P. Sommer, F. Schellroth, M. Fischer, J. Schlechtendahl, Message-oriented middleware for industrial production systems, *in 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich* (2018), pp. 1217–1223
23. 5G EVE: first implementation of the interworking reference model. Deliverable D3.3 (2019). https://doi.org/10.5281/zenodo.3628179
24. L. Magnoni, Modern messaging for distributed systems. J. Phys. Conf. Ser. **608**, 012038 (2015). https://doi.org/10.1088/1742-6596/608/1/012038
25. Beats. https://www.elastic.co/beats/. Accessed 26 Feb 2021
26. ELK Stack. https://www.elastic.co/what-is/elk-stack. Accessed 26 Feb 2021
27. 5G EVE: experimentation tools and VNF repository. Deliverable D4.1 (2019). https://doi.org/10.5281/zenodo.3628201
28. 5G EVE: first version of the experimental portal and service handbook. Deliverable D4.2 (2019). https://doi.org/10.5281/zenodo.3628316
29. 5G EVE: report on benchmarking of new features and on the experimental portal (2nd version). Deliverable D4.4 (2020). https://doi.org/10.5281/zenodo.3946283

30. B. Nogales, I. Vidal, D.R. Lopez, J. Rodriguez, J. Garcia-Reinoso, A. Azcorra, Design and deployment of an open management and orchestration platform for multi-site NFV experimentation. IEEE Commun. Mag. **57**(1), 20–27 (2019)
31. 5G EVE: initial pilot test and validation. Deliverable D2.4 (2020). https://doi.org/10.5281/zenodo.3946365
32. Ubuntu. https://ubuntu.com/. Accessed 26 Feb 2021
33. Proxmox. https://www.proxmox.com/en/. Accessed 26 Feb 2021
34. K3s. https://k3s.io/. Accessed 26 Feb 2021
35. 5G EVE Monitoring Dockerized Environment. https://github.com/5GEVE/monitoring_dockerized_environment. Accessed 26 Feb 2021
36. Sangrenel. https://github.com/jamiealquiza/sangrenel. Accessed 26 Feb 2021
37. Python. https://www.python.org/. Accessed 26 Feb 2021
38. Apache Kafka: allow consumers to fetch from closest replica. KIP-392 (2019). https://cwiki.apache.org/confluence/display/KAFKA/KIP-392%3A+Allow+consumers+to+fetch+from+closest+replica
39. Node.js. https://nodejs.org/en/. Accessed 26 Feb 2021
40. Apache ZooKeeper. https://zookeeper.apache.org/. Accessed 26 Feb 2021
41. Docker. https://www.docker.com/. Accessed 26 Feb 2021
42. P. Dobbelaere, K.S. Esmaili, Kafka versus RabbitMQ: a comparative study of two industry reference publish/subscribe implementations: industry paper, in *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems. DEBS '17* (Association for Computing Machinery, New York, NY, USA, 2017), pp. 227–238. https://doi.org/10.1145/3093742.3093908
43. W. Nakimuli, G. Landi, R. Perez, M. Pergolesi, M. Molla, C. Ntogkas, G. Garcia-Aviles, J. Garcia-Reinoso, M. Femminella, P. Serrano, F. Lombardo, J. Rodriguez, G. Reali, S. Salsano, Automatic deployment, execution and analysis of 5G experiments using the 5G EVE platform, *in 2020 IEEE 3rd 5G World Forum (5GWF)* (2020), pp. 372–377
44. 5G EVE Github repository. https://github.com/5GEVE. Accessed 26 Feb 2021

Publisher's Note