A Distributed Framework Based on Publish-Subscribe to Monitor Beyond 5G Networks

Ramon Perez (✉ ramon.perez@telcaria.com)

Telcaria Ideas S.L. <https://orcid.org/0000-0003-3450-6141>

Jaime Garcia-Reinoso

Universidad Carlos III de Madrid

Aitor Zabala

Telcaria Ideas S.L.

Pablo Serrano

Universidad Carlos III de Madrid

Albert Banchs

Universidad Carlos III de Madrid

Research

Keywords: monitoring, data collection, 5G experimental validation, 5G trials, 5G multi-site platform, publish-subscribe paradigm, Beyond 5G networks

Posted Date: October 27th, 2020

DOI: <https://doi.org/10.21203/rs.3.rs-96351/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at EURASIP Journal on Wireless Communications and Networking on April 7th, 2021. See the published version at <https://doi.org/10.1186/s13638-021-01962-y>.

RESEARCH

A Distributed Framework Based on Publish-Subscribe to Monitor Beyond 5G Networks

Ramon Perez^{1,2*}, Jaime Garcia-Reinoso², Aitor Zabala¹, Pablo Serrano² and Albert Banchs^{2,3}

This paper is an extension of the following paper: [1].

*Correspondence:

ramon.perez@telcaria.com

¹ Telcaria Ideas S.L., C/

Barrionuevo, 8, 28911, Leganes,
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 exists the necessity of a flexible and efficient monitoring system, 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 scope, current monitoring solutions are limited, as they have to be able to maintain 5G performance in a distributed system with heterogeneous resources and still be efficient and sustainable. 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, to confirm that it suits the requirements of the proposed scenarios, and discuss how the architecture could be mapped to other 5G or Beyond 5G 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 standardisation, 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 Key Performance Indicators (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. In this way, 5G EVE [4] is a European project that 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 [5]), the project presented in [6] the proposed architecture and the main innovation areas addressed, including the KPI Framework for performance diagnosis.

One of the main components of this architecture related to the aforementioned innovative topics is the Monitoring service, which is intended to collect all the metrics generated by the different elements involved in an experiment to show their evolution over time to the experimenter, and to feed such data to the KPI Validation Framework to confirm the achievement of the KPIs, or also enabling new workflows like the optimization of network performance.

This paper presents the Monitoring framework defined by the 5G EVE project, which has been designed to be flexible enough to accommodate a variety of use cases. In fact, the framework not only supports the heterogeneous set of verticals participating in the project (which include use cases that range from Connected Industry 4.0 to gaming), but also verticals participating in related initiatives (as described in Section 2), aiming at providing an adaptable platform that could fit and scale in different network deployments, anticipating 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, 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 re-configurable architecture.
- Section 4 justifies and details the implementation selected by 5G EVE to instantiate the proposed architecture, based on the publish-subscribe paradigm.
- Section 5 validates such implementation against the requirements imposed to the Monitoring architecture from the 5G EVE project specifications, 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, Section 7 concludes the paper and presents our future work.

2 Related Work

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

First of all, this Monitoring platform has been designed and implemented within the scope of an European project related to the research on 5G networks. While a number of other 5G projects (European and International) have addressed monitoring functionalities, limited work in this context have 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) [7], which is an extension of the monitoring solution already proposed in the 5G-TRANSFORMER project [8].

Another context that is present in these environments 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) [9] and the Management Data Analytics Function (MDAF) [10] for 5G networks.

Other organizations, such as the O-RAN alliance, also contemplates similar components in their architectures [11], and ETSI has also defined comparable assisting elements within the Industry Specification Groups (ISGs) on Experiential Networked Intelligence (ENI) and Zero touch network & Service Management (ZSM) [12]. Moreover, open-source initiatives such as ONAP [13] 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, 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 [14] or [15], 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 [16], 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

The 5G EVE complete platform, 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 behaviour of these components. In this sense, a general-purpose Monitoring architecture is desired, so that it can also fit in other similar scenarios to the 5G EVE one, with regard to multi-stakeholder environments.

To start with the definition of this Monitoring architecture, a thorough analysis of the 5G EVE infrastructure and service requirements [5] has been taken into account in order to extract a set of preliminary characteristics to be envisioned for the Monitoring service. These are the following:

- 1 The Monitoring distribution architecture must support multi-site experiments involving distant sites.
- 2 The platform must deal with experiments that may generate monitoring data in the order of gigabytes.
- 3 Monitoring data has to be available to experimenters after the experiment 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, also defined within the 5G EVE project, 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 Figure 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 *experiment infrastructure* to be monitored, included here for the sake of completeness, and which may be User Equipment devices

(UEs), monitoring tools, (4G or 5G) radio antennas, Physical Network Functions (PNFs) or Virtual Network Functions (VNF).

- In light blue, the elements that compose the Monitoring platform itself, which will be presented next by following a bottom-up/west-east approach.

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 experiment's 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.

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 Figure 1.

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 Figure 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 the 5G EVE format [17] to satisfy constraint (6) presented before. This might require a translation from a proprietary or different standard formats to the 5G EVE one, in order to handle all the messages received from the *MEFs* in a unified way.

The monitoring data is 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 a unified way, preserving the data privacy of each site.
- The **Inter-site broker**, which interconnects all sites together to both:
 - 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 ten seconds 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.

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-experiment basis, also enabling the necessary security mechanisms to ensure that only the actors belonging to a given experiment can manage the monitored data of their experiment and not others.

4 Implementation Based on the Publish-Subscribe Paradigm

Taking into account the general-purpose Monitoring architecture described in Section 3, the 5G EVE project extended it to build its own Monitoring platform. 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, the implementation of the Monitoring architecture presented in Section 3 over the 5G EVE architecture [4] [18] results in the composition of a specific component chain, depicted in Figure 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 [19].

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 Figure 2, each *Intra-site broker* is represented by a **Site Broker** entity, and the *Inter-site broker*, together with the *Metrics Management* service, are implemented by the **Data Collection Manager** component in 5G EVE architecture.

All the *Site Brokers* and the *Data Collection Manager* are based on *Apache Kafka* [20], 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

[21]. 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 [22]. This makes *Kafka* an optimal solution for data-movement, frequently adopted as pipe to different processing systems [23].

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 towards 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 Section 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 [22]. 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 5G EVE format mentioned in Section 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.
- Signalling topics, used to deliver the name of data topics related to each metric or KPI to be monitored for a given experiment. 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 experiment execution, 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 experiment), 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* [24] 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 pub-

lished 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* entities with a solution based on the *Elastic (ELK) Stack* [25] [26]. 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 [27] [28]:

- 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 experiment thanks to *Elasticsearch*, also from the *Elastic Stack*.
- The *Data Visualization* component, in charge of enabling the monitoring of the progress of the experiment in terms of that monitoring data displayed through *Kibana* from the *Elastic Stack*. For this purpose, a set of dashboards are created for experiment, presenting the graphs related to each metric or KPI monitored.

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, starting with **single-broker experiments** to characterize the platform in terms of several performance parameters, and finishing with **multi-broker experiments** to check the impact of having the two brokering levels described in Section 3.

5.1 System Assumptions

The definition of the System Under Test (SUT) parameters is bound to the 5G EVE multi-site platform's operation, where a set of experiments 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 experiments. As the 5G EVE project initially proposes the validation of experiments from six specific use cases [5], the execution of an experiment from each use case at the same time can be taken as the worst case study to validate, resulting in six simultaneous 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 experiment execution, depending on vertical's needs. For this evaluation process, as these metrics can be extracted from different sources (*e.g.*, UEs, VNFs, PNFs, etc.), and each source may have several related metrics or KPIs, 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^[1] 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.

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 Section 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) [29] deployed in a server located in the 5G EVE Spanish site facility, 5TONIC [30], using *Proxmox* [31] as virtualization environment, and *K3s* (a lightweight *Kubernetes* distribution) [32] to orchestrate the containerized components^[2] deployed in each VM. This server is equipped with 40 Intel(R) Xeon(R) CPU E5-2630 v4 at 2.20GHz and 128 GB RAM. The distribution of components in each VM can be seen in Figure 3.

The proposed scenario intends to simulate the monitoring and data collection process of the metrics and KPIs related to a set of experiments. 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 towards the *Data Collection Manager*, placed in the

^[1]This figure does not include the signalling topics presented in Section 4, whose footprint is not significant compared to these data topics.

^[2]The images of these components can be found in [33].

Figure 3 Testbed architecture.

Kubernetes Master node. Regarding the *Data Shippers*, this role is played by two components:

- *Sangrenel* [34], 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* [35], 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*^[3] [37], 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 to 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 Figure 2 have

^[3]This programming language has been used in order to make use of *Kafka*'s KIP-392 feature, to receive data from the closest replica [36].

been implemented with a solution based on *Apache Kafka* and the *Elastic Stack*. A *ZooKeeper* [38] 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 has been previously replicated from the *Site Broker*).

For monitoring the resource consumption of each container, *Docker* [39] 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 Figure 3 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 Section 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 to 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.

This assumption is confirmed in Figure 4 for the batch write latency analysis in one experiment with multiple topics, according to the per-experiment topic distribution described in Section 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 Figure 4, 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: (i) the number of topics created and running in the system as concurrent processes, due to the execution of simultaneous experiments, and (ii) 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 if the superposition property can be applied when both parameters are modified simultaneously. For doing this, the study was divided in 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 if the superposition of individual effects is present.

Part (1) is presented in Figure 5, where the CPU consumption and the batch write latency related to 100 B aggregated data traffic^[4] are evaluated for different examples of experiments:

^[4]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 Figure 4.

- 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 experiments is conserved, which is achieved by dividing the message rate aforementioned by the number of experiments deployed.

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 experiments (*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 Figure 6, related to the part (2) of the study aforementioned.

Here, when the number of experiments 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 Figure 6 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 [40], 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 the deployment of a second experiment 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), Figure 7 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 Figure 7); 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 Figure 7). 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 Figures 6 and 7, the direct solution is to build mechanisms and processes that allow system scalability, mainly oriented to the 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 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 8 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 full multi-broker platform, as built in the testbed already presented in Figure 3, will be evaluated in terms of the performance parameters already presented in Section 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 Figure 7.

In this case, the meaning of experiment will be a bit different. This way, each experiment deployed in multi-broker experiments will be executed in a particular *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 Section 5.1 (*i.e.*, 10,24 *Mbps*), which is the throughput hard limit to avoid saturation, as stated in Figure 7.

5.4.1 Impact on Latency

The first performance parameter to be evaluated is the latency, in the different acceptations that were defined in Section 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 Figure 9. Here, a similar effect than the one obtained in Figure 4 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 Figure 7 can be seen also here: the latency does not vary even though the total throughput received by the Monitoring platform increases due to the deployment of new experiments.

Figure 9 Evolution of the three types of latency in multi-broker experiments.

Furthermore, the results^[5] 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 colour 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 colour 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 great impact on latency, as it is increased to nearly *ms* in the worst case. And finally, the highest value on latency (approx. 2,5–2,6 seconds) is obtained for the end-to-end latency (the lighter colour 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 Figure 7, where the packet loss increases with the increase of the total throughput received in the platform. This effect can be seen in Figure 10, where the performance results from different brokers have been aggregated due to the results obtained in Section 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 7 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.

^[5]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 take into account.

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 on Section 5 have 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 experiments, 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 16000 and 32000 packets per second, this exponential growth is stopped and the I/O message rate falls 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 towards 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 deploying several experiments, 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 experiments 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.

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 experiments in the Monitoring platform [41] or the enhanced implementation of the Monitoring platform, which is available in the 5G EVE Github repository [42].

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 Section 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 [33], 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.

Methods/Experimental

Although the performance evaluation process is fully detailed in Section 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 experiments, in which only the first level of brokering of the architecture is tested, to multi-broker experiments, where the full clustered 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 experiments.

Each experiment 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 minutes, and the data obtained for each performance parameter can be analyzed by using statistical measures like the mean, variance and standard deviation.

List of Abbreviations

2G: Second-Generation; 3GPP: 3rd Generation Partnership Project; 4G: Fourth Generation; 5G: Fifth-Generation; 5G-VoMS: 5G-Vertical-oriented Monitoring System; 6G: Sixth-Generation; ACK: Acknowledgement; ACL: Access Control List; AI: Artificial Intelligence; B: Byte; CPU: Central Processing Unit; ELK: Elasticsearch Logstash Kibana; eMBB: Enhanced Mobile Broadband; ENI: Experiential Networked Intelligence; ETSI: European Telecommunications Standards Institute; GB: Gigabyte; GHz: Gigahertz; I/O: Input/Output; IoT: Internet of Things; ISG: Industry Specification Groups; KB: Kilobyte; KPI: Key Performance Indicator; 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; 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 & Service Management.

Availability of Data and Materials

The datasets used and/or analysed during the current study are available from the corresponding author on reasonable request.

Competing Interests

The authors declare that they have no competing interests.

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). The paper solely reflects the views of the authors. The Commission is not responsible for the contents of this paper or any use made thereof.

Author's Contributions

RP was the main responsible for the design, implementation and evaluation of the Monitoring platform presented in this paper, JGR contributed in the design and implementation of the Monitoring architecture, AZ contributed in 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.

Acknowledgements

Not applicable.

Author's Information

Not applicable.

Author details

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

References

- Perez, R., Garcia-Reinoso, J., Zabala, A., Serrano, P., Banchs, A.: A monitoring framework for multi-site 5g platforms. In: 2020 European Conference on Networks and Communications (EuCNC), pp. 52–56 (2020)
- Elayoubi, S.E., Fallgren, M., Spapis, P., Zimmermann, G., Martín-Sacristán, D., Yang, C., Jeux, S., Agyapong, P., Campoy, L., Qi, Y., *et al.*: 5G service requirements and operational use cases: Analysis and METIS II vision. In: 2016 European Conference on Networks and Communications (EuCNC), pp. 158–162 (2016). IEEE
- Foukas, X., Patounas, G., Elmokashfi, A., Marina, M.K.: Network slicing in 5G: Survey and challenges. IEEE Communications Magazine **55**(5), 94–100 (2017)
- Gupta, M., Legouable, R., Rosello, M.M., Cecchi, M., Alonso, J.R., Lorenzo, M., Kosmatos, E., Boldi, M.R., Carrozzo, G.: The 5G EVE End-to-End 5G Facility for Extensive Trials. In: 2019 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–5 (2019). IEEE
- 5G EVE: Requirements definition and analysis from participant vertical industries. Deliverable D1.1 (2018). <https://doi.org/10.5281/zenodo.3530391>
- 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>
- Papagianni, C., Manges-Bafalluy, J., Bermudez, P., Barmounakis, S., De Vleeschauwer, D., Brenes, J., Zeydan, E., Casetti, C., Guimarães, C., Murillo, P., Garcia-Saavedra, A., Corujo, D., Pepe, T.: 5growth: Ai-driven 5g for automation in vertical industries. In: 2020 European Conference on Networks and Communications (EuCNC), pp. 17–22 (2020)
- de la Oliva, A., Li, X., Costa-Perez, X., Bernardos, C.J., Bertin, P., Iovanna, P., Deiss, T., Manges, J., Mourad, A., Casetti, C., Gonzalez, J.E., Azcorra, A.: 5g-transformer: Slicing and orchestrating transport networks for industry verticals. IEEE Communications Magazine **56**(8), 78–84 (2018)
- 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>
- 3GPP: Management and orchestration; Architecture framework. TS 28.533, v16.2.0 (2019). <https://www.3gpp.org/DynaReport/28533.htm>
- Alliance, O.-R.: O-RAN: Towards an Open and Smart RAN. White Paper (2018)
- ETSI: Network Transformation; (Orchestration, Network and Service Management Framework). White Paper No. 32 (2019)
- 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>
- Javed, A., Hejanko, K., Buda, A., Främling, K.: CEFIoT: A fault-tolerant IoT architecture for edge and cloud. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 813–818 (2018)
- Martín, C., Garrido, D., Díaz, M., Rubio, B.: From the Edge to the Cloud: Enabling Reliable IoT Applications. In: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 17–22 (2019)
- Yuan, Q., Ji, X., Tang, H., You, W.: Toward Latency-Optimal Placement and Autoscaling of Monitoring Functions in MEC. IEEE Access **8**, 41649–41658 (2020)
- 5G EVE: Second implementation of the interworking reference model. Deliverable D3.4 (2020). <https://doi.org/10.5281/zenodo.3946323>
- Garcia-Reinoso, J., Roselló, M.M., Kosmatos, E., Landi, G., Bernini, G., Legouable, R., Contreras, L.M., Lorenzo, M., Trichias, K., Gupta, M.: The 5G EVE Multi-site Experimental Architecture and Experimentation Workflow. In: 2019 IEEE 2nd 5G World Forum (5GWF), pp. 335–340 (2019). IEEE
- 5G EVE: Interworking Reference Model. Deliverable D3.2 (2019). <https://doi.org/10.5281/zenodo.3625689>
- Apache Kafka. <https://kafka.apache.org/>. Last Accessed: 14 October 2020
- Sommer, P., Schellroth, F., Fischer, M., Schlechtendahl, J.: Message-oriented Middleware for Industrial Production Systems. In: 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, pp. 1217–1223 (2018)
- 5G EVE: First implementation of the interworking reference model. Deliverable D3.3 (2019). <https://doi.org/10.5281/zenodo.3628179>
- Magnoni, L.: Modern Messaging for Distributed Systems. Journal of Physics: Conference Series **608**, 012038 (2015). doi:10.1088/1742-6596/608/1/012038
- Beats. <https://www.elastic.co/beats/>. Last Accessed: 14 October 2020
- ELK Stack. <https://www.elastic.co/what-is/elk-stack>. Last Accessed: 14 October 2020
- 5G EVE: Experimentation tools and VNF repository. Deliverable D4.1 (2019). <https://doi.org/10.5281/zenodo.3628201>
- 5G EVE: First version of the experimental portal and service handbook. Deliverable D4.2 (2019). <https://doi.org/10.5281/zenodo.3628316>
- 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>
- Ubuntu. <https://ubuntu.com/>. Last Accessed: 14 October 2020
- Nogales, B., Vidal, I., Lopez, D.R., Rodriguez, J., Garcia-Reinoso, J., Azcorra, A.: Design and deployment of an open management and orchestration platform for multi-site nfv experimentation. IEEE Communications Magazine **57**(1), 20–27 (2019)
- Proxmox. <https://www.proxmox.com/en/>. Last Accessed: 14 October 2020
- K3s. <https://k3s.io/>. Last Accessed: 14 October 2020
- 5G EVE Monitoring Dockerized Environment.

- https://github.com/5GEVE/monitoring_dockerized_environment. Last Accessed: 14 October 2020
34. Sangrenel. <https://github.com/jamiealquiza/sangrenel>. Last Accessed: 14 October 2020
 35. Python. <https://www.python.org/>. Last Accessed: 14 October 2020
 36. 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>
 37. Node.js. <https://nodejs.org/en/>. Last Accessed: 14 October 2020
 38. Apache ZooKeeper. <https://zookeeper.apache.org/>. Last Accessed: 14 October 2020
 39. Docker. <https://www.docker.com/>. Last Accessed: 14 October 2020
 40. Dobbelaere, P., Esmaili, K.S.: 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, pp. 227–238. Association for Computing Machinery, New York, NY, USA (2017). doi:10.1145/3093742.3093908
 41. Nakimuli, W., Landi, G., Perez, R., Pergolesi, M., Molla, M., Ntogkas, C., Garcia-Aviles, G., Garcia-Reinoso, J., Femminella, M., Serrano, P., Lombardo, F., Rodriguez, J., Reali, G., Salsano, S.: Automatic deployment, execution and analysis of 5g experiments using the 5g eve platform. In: 2020 IEEE 3rd 5G World Forum (5GWF), pp. 372–377 (2020)
 42. 5G EVE Github repository. <https://github.com/5GEVE>. Last Accessed: 14 October 2020

Figures

Figure 1
Monitoring metrics architecture.

Figure 2
Component chain that implements the general Monitoring metrics architecture in the 5G EVE platform.

Figure 3

Testbed architecture.

Figure 4

Batch write latency distribution in one experiment with 20 topics.

Figure 5

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

Figure 6

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

Figure 7

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

Figure 8

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

Figure 9

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

Figure 10

Evolution of Data Collection Manager CPU consumption and I/O message rate in multi-broker experiments.