Meeting the Observability Challenges for VNFs in 5G systems

Wolfgang John, Farnaz Moradi
Ericsson Research, Stockholm, Sweden
email: {wolfgang.john,farnaz.moradi}@ericsson.com

Bertrand Pechenot, Pontus Sköldström
Acreo Swedish ICT, Stockholm, Sweden
email: {berpec,ponsko}@acreo.se

***Abstract*—5G mobile communication systems will need to accommodate a variety of use-cases, resulting in a diverse set of requirements. To meet these requirements, 5G systems take advantage of modern virtualization possibilities offered by Network Function Virtualization (NFV), enabling deployment agility and dynamicity of virtualized network functions. With the transformation of telecom towards virtualized environments, advanced observability possibilities gain increasing importance as one of the essential prerequisites, especially for successful DevOps operations. However, deployment agility also puts specific requirements on monitoring solutions in order to adapt automatically and continuously to frequent changes in service deployments. In this short-paper, we establish and discuss essential properties of observability systems for virtual network functions in a 5G context. We take these properties as guiding design principles for our software-defined monitoring framework and outline how to evolve our existing components towards a flexible, scalable, and programmable observability solution for microservice-based NFV with features for increased manageability.**

I. Introduction

5th generation mobile communication systems (5G) will be designed to accommodate a variety of use-cases (mobile broadband, media delivery, mission-critical, etc.), resulting in a diverse set of requirements - like vast numbers of connected devices, high capacities, and low latencies [1]. To facilitate 5G as a single, common platform realizing this wide range of use-cases, flexibility, agility, and dynamicity are important implicit system requirements. An essential part of the 5G system design is thus the virtualization of wireless networks[1], allowing to flexibly adjust placement and scale of virtual network functions (VNF) to specific use-case requirements.

The most promising technology for virtualization of network/service functions is Network Function Virtualization (NFV)[2], which allows decoupling of well isolated service instances from infrastructure resources (Fig.1). Flexibility and dynamicity in NFV is further increased when VNFs are realized by containers and run with DevOps-style continuous integration and deployment. Using containers instead of VMs opens up for efficient and resilient service applications realized by a microservice architecture [2], i.e., independently deployable containers communicating with each other via APIs.

Monitoring of services and service components has historically been of importance for telecom operators for network troubleshooting and to follow up on customer service level agreements (SLAs). With the transformation of telecom towards virtualized environments, real-time observability gains even more importance as one of the key prerequisites for DevOps operations [3]. Assuming that some 5G applications will be realized by microservices, corresponding VNF components (VNFCs), e.g., packaged as containers, can be started, stopped, updated, scaled, and migrated frequently. A monitoring system thus should be able to adapt automatically and continuously to the changes in microservices deployments and communications, and observe metrics related to both containers as well as network and compute infrastructure. As a result, we conclude that existing monitoring frameworks will need to evolve in order to fulfill the following properties:

- *Flexibility*: observability needs to be ensured irrespective of the virtualization format and technology (e.g., Virtual Machines (VMs) or containers in various platforms).
- *Dynamicity*: observability components need to follow the dynamic behavior of microservice instances seamlessly (i.e., frequent scaling and migration of VNFCs).
- *Scalability and non-intrusiveness*: an observability system needs to scale to large numbers of services/VNFCs without a significant footprint on resource utilization.
- *Programmability*: observability features need to be programmable in order to enable full automation and tight integration with orchestration and policy systems.
- *Manageability*: an observability system should be easy to manage, e.g., by providing single-touch management and configuration of monitoring functions.
- *Decentralization*: observability components need to deal with distributed telco systems spreading over very large geographical areas, in contrast to typical IT datacenters with well connected, locally centralized resources.
- *Multi-tenant isolation*: an observability system needs to support isolation of tenants to cater for the wide range of use-cases supported in parallel by one 5G platform.
- *Timeliness*: observability results need to be provided in close-to real-time to both dev and ops components and personnel, facilitating fast control-loops.

This paper presents a software-defined monitoring framework for microservice-based NFV, designed with the above requirements in mind. In our framework, monitoring intents realize high-level programmability, resulting in automatic instantiation and configuration of monitoring components following the dynamic behavior of service VNFCs. Locally aggregated results are shared by a carrier-grade messaging system.

This paper is organized as follows: Section II discusses related work; Section III presents our proposed framework and ongoing work. Finally, Section IV concludes the paper.

[1]We consider the wireless network to include both radio access network (RAN) functions as well as core network (CN) functions.

[2]http://www.etsi.org/technologies-clusters/technologies/nfv

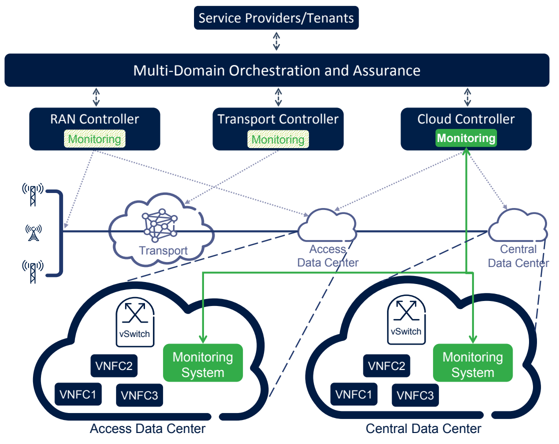

Fig. 1. 5G multi-domain orchestration and assurance architecture with radio, transport, and Cloud domains (more details in [4]). RAN functionality is split between remote radio units and vRAN functions in access data centers, while core network (CN) functions are running in a central data center. The focus of this paper is the Cloud/DC monitoring system (depicted in solid green).

II. Related Work

There are a variety of different observability systems that have been designed for monitoring in virtualized environments. However, these solutions do not provide all the properties required for monitoring VNF instances in 5G systems.

In OpenStack Ceilometer is the service responsible for metering data collection and Monasca[3] provides a scalable, multi-tenant, and fault-tolerant monitoring-as-a-service solution. Although these solutions are widely used, they do not satisfy requirements such as flexibility and manageability.

Monitoring of containers can be performed using a variety of tools. For instance, Docker provides a stats API for obtaining a live stream of CPU, memory, network I/O, and block I/O metrics. Other tools such as cAdvisor[4] and Prometheus[5] are also widely used for monitoring resource usage of containers. In Kubernetes[6], Heapster[7] discovers all the nodes in the cluster and queries for container resource usage information and transfers aggregate data to a configurable backend for storage and visualization. These tools do not provide the means for monitoring the network performance between containers forming microservices and have limitations in satisfying requirements such as flexibility.

JCatascopia [5] is a monitoring system which dynamically and automatically deploys monitoring agents on physical/virtual instances. The agents manage probes, which collect low-level monitoring data, and aggregate and transfer metrics to monitoring servers via a pub-sub messaging system. JCatascopia also uses filtering capabilities to minimize both storage and communication overhead. Although JCatascopia addresses many challenges related to 5G observability, it has limitations with respect to flexibility and multi-tenancy support.

NFVPerf [6] is introduced for performance analysis and bottleneck identification in NFV. NFVPerf obtains a *VNF forwarding graph* as input from the cloud management system and then configures port mirroring to capture packets on all hops of the VNF forwarding chain for measuring e.g., per-hop delay. The collected metrics are transferred to a central analysis module, where application performance degradations are identified and localized. NFVPerf does not fulfill requirements such as dynamicity, manageability, and programmability.

The T-Nova projectintroduced an NFV monitoring framework [7] which gathers monitoring data from OpenStack and OpenDaylight as well as monitoring agents running inside each VNF VM. The metrics are collected and aggregated by a monitoring manager which produces alarms/events towards the orchestrator. This framework focuses on addressing monitoring challenges related to scalability and alarm generation. However, T-Nova has a focus on OpenStack VMs and does not introduce programmability features.

As part of the SP-DevOps concept [8], the UNIFY project[8] developed a solution for efficient and programmable observability of software-defined environments. The SP-DevOps observability process takes advantage of programmable monitoring functions embedded in the infrastructure to provide monitoring results with a fraction of the overhead of typical management tools. Monitoring functions are deployed programmatically and provide data through a scalable messaging system, easing integration with 3rd party management and orchestration components. While this process meets most monitoring requirements of 5G, it has shortcomings with respect to manageability and integrated programmability [9].

III. Software-Defined Monitoring Framework

In this paper we present our software-defined monitoring framework which aims to fulfill the desired properties for monitoring of 5G systems as outlined in Section I. The framework adapts the UNIFY SP-DevOps observability process to a 5G scenario and advances it with updated programmability and autonomous configuration features.

The architecture of our framework is shown in Fig. 2, and represents the monitoring system for the Cloud/DC part of a 5G network (Fig. 1). In this framework, distributed Monitoring Controllers (MC) are responsible for management of monitoring and analytics functions. ① The MCs identify the type of the instantiated VNF containers and ② obtain the corresponding monitoring intents from the MEASURE repository. ③ The container management systems obtain container images of the required Monitoring and Analytics Functions (MFs and AFs) from the Monitoring function repository, and ④ deploy the MF and AF containers on physical/virtual hosts. ⑤ Once MFs and AFs are configured, the collected monitoring results or alarms are transferred to a monitoring management system via a flexible messaging system called DoubleDecker.

This solution provides programmability and flexibility by through MEASURE monitoring intents (Section III-A). Dis-

[3]http://monasca.io/
[4]https://github.com/google/cadvisor
[5]https://prometheus.io/
[6]http://kubernetes.io/
[7]https://github.com/kubernetes/heapster

[8]https://www.fp7-unify.eu/

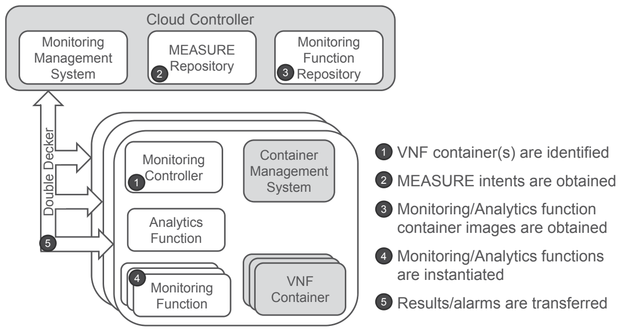
Cloud Controller
Monitoring Management System
MEASURE Repository
Monitoring Function Repository
Double Decker
Monitoring Controller
Container Management System
Analytics Function
Monitoring Function
VNF Container
1 VNF container(s) are identified
2 MEASURE intents are obtained
3 Monitoring/Analytics function container images are obtained
4 Monitoring/Analytics functions are instantiated
5 Results/alarms are transferred

Fig. 2. Software-defined monitoring framework

tributed MCs realized within the container execution environment further ensure that dynamic and autonomous management is achieved (Section III-B). Finally, the DoubleDecker messaging system allows the system to operate flexible and decentralized, while additionally adding scalability features and multi-tenancy support for 5G observability (Section III-C).

A. Programmable monitoring deployment

To allow higher layer orchestration functions and tenants to programmatically define their monitoring needs in a flexible way we adapted the MEASURE monitoring intents, originally designed in the UNIFY project [8]. These intents focus on what metric to monitor (rather than on how to configure a particular MF), and where in the service to monitor it (rather than which specific container or network interface), and how to analyze the metrics in AFs. This lets the MC decide which available monitoring and analytics functions to use, how to configure them, and where to instantiate them. Tenants and higher layers need not know any details of the infrastructure, but simply describe the metrics they want, as shown below:

```
monitor:
  m1 = cpu(vnfc1); m2 = latency(vnfc1,vnfc2);
zones:
  z1 = m1 < 0.8 AND m2 < 10ms; z2 = !z1;
reactions:
  z2->z1 = publish(alarm, "Bad");
  z1->z2 = publish(alarm, "Good");
```

In the **monitor** section, the required metrics are described together with a reference to the target entity. Each metric is assigned to a variable that can be used to refer to the monitoring results. In this example the CPU load of the VNF container `vnfc1` and the latency between containers `vnfc1` and `vnfc2` are requested and assigned to `m1` and `m2`. **Zones** describe how to calculate different states based on the monitoring results. Here, state `z1` is defined as low CPU load and low latency, and state `z2` as the negation of `z1`. In the zone definitions aggregation functions supported by the Analytics Function can be called, e.g. for calculating a moving average on a metric. Finally, **reactions** describes actions that should be taken when moving between states, for example publishing a message when going between zones `z1` and `z2`.

This information, together with data from the MEASURE and Monitoring Function repositories (see Fig. 2), is used by the MC to resolve the abstract entities (mapping e.g. `vnfc1` to a particular Docker container), and to generate configuration for the Analytics Function, programming it to perform the zone calculations and actions. The repositories hold information about available metrics, where they can be applied, and which Monitoring Functions can provide them. They also hold information about the MFs themselves, e.g., how to start and configure them.

B. Autonomous monitoring management

In our framework, a container-based monitoring system (ConMon) is used for monitoring of microservices [10]. ConMon has a decentralized architecture and enables multi-tenancy by separating monitoring components from microservice instances. It provides autonomous management of MFs/AFs and dynamically adapts to changes in the virtualization environment and communication between microservice instances in a timely manner. Although ConMon is realized for containers, its generic design makes it suitable for monitoring in other virtual execution environments as well (e.g., VMs).

The core components of ConMon are Monitoring Controllers (MCs) which are distributed across physical servers in a datacenter (see Fig. 2). Each MC listens to life-cycle events of the service containers (i.e., VNF containers) generated by the container management system (e.g., Docker). Whenever a VNF container which requires monitoring is started, the MC immediately instantiates and configures the needed MFs/AFs.

In ConMon, MFs are instantiated as separate containers that run adjacently to the service containers. This means that there is no need for executing a monitoring process inside each service container and allows dynamic programming of monitoring components without adversely affecting the service applications. Such a monitor container can be used for monitoring multiple VNFCs which belong to the same tenant or for infrastructure-level monitoring of the VNFCs that belong to different tenants, e.g., round-trip-time (RTT) measurements.

The information about the metrics that need to be monitored are derived from monitoring intents. In our framework we use MEASURE intents as described in Section III-A. Once an MC identifies the type of VNFC being instantiated locally, e.g., by inspecting the VNF container, it obtains the corresponding MEASURE intents from the MEASURE repository. The MC then instantiates and configures MFs for the specified metrics and configures the AFs based on the *zones* definitions.

An autonomous monitoring framework should keep up with the dynamic nature of containers in microservice architectures and adapt the monitoring accordingly. ConMon does so by dynamically setting up and configuring network performance monitoring between containers. Each MC instantiates a passive monitoring container and configures the virtual switch (e.g., OpenvSwitch) to mirror the traffic to/from the VNF containers towards it. The MF which is executed inside the passive monitoring container can identify the source and destination address of the VNF containers which communicate with each other, e.g., `vnfc1` and `vnfc2`. The local MC will then discover the remote MC and exchange control messages. Once the MCs have identified each other, they can start monitoring sessions, e.g., RTT measurements. In this case both MCs instantiate and configure MFs and start monitoring sessions. More details about ConMon and the evaluation results indicating its feasibility and effects of passive monitoring on the service and background traffic can be found in [10].

C. Flexible, scalable, and isolated data transport

The UNIFY project developed a system that provides carrier-grade and scalable messaging for easy integration of diverse monitoring and orchestration functions as well as efficient and secure transport of results [11]. DoubleDecker[9] is a messaging system with distributed brokers allowing interconnection between all components in our framework (see Fig. 2), acting as DoubleDecker clients. The main features of DoubleDecker in this context are the possibility to dynamically connect and disconnect components, the distributed broker architecture, and multi-tenancy isolation.

The first feature is critical because the framework will autonomously adapt to VNF dynamics by continuous instantiation, connection, and disconnection of respective MFs/AFs. functions. The messaging system is thus built to be tolerant to disconnection with a built-in reconnection procedure. Keeping track of the connected clients is also an important feature as it allows better fault tolerance. Furthermore, DoubleDecker is a simple to use and technology-agnostic transport mechanism between components realized in different implementation languages and execution environments.

Secondly, the distributed architecture is a key feature, and the main driver behind our decision to use this system instead of other popular messaging queues. Brokers can be distributed on each physical or virtual node with a limited memory footprint. Local clients can communicate with each other without using any resources external to the node hence limiting network load. Decentralized brokers also improve robustness - if the connection between two brokers is lost, they can still operate independently and will reconnect once connectivity is restored. This fits perfectly with the ConMon system which is also able to take local decisions even if the communication with control/orchestration layers is unavailable.

Finally, all DoubleDecker clients have to provide a tenant specific public/private key-pair on registration. This key-pair is then used to encrypt messages sent though the system, ensuring isolation of messages from different tenants in a transparent way. Such multi-tenant isolation is crucial to allow multiple use-cases in parallel on a single 5G system.

D. Discussion and work in progress

We presented a system that integrates novel components to form a complete observability solution ready to meet the challenges of NFV-based 5G systems. However, to fully support all observability properties identified in this paper, we are still refining the framework to facilitate tighter integration of the monitoring system with the control/orchestration architecture (cf. Fig.1). This is important to synchronize life-cycles of both VNFC instances and monitoring functions and handle resource dependencies. Currently, we are also adding features to further simplify manageability, such as auto-naming/scoping for pub/sub messaging, and key management in DoubleDecker.

[9]Code available: https://github.com/Acreo/DoubleDecker

As future direction, we will continue the ongoing implementation of a test-bed in a local datacenter, realizing the complete software-defined monitoring system by fully integrating the components described earlier. Our initial experiments led to valuable insights, and we expect further learnings once our tests evolve. As upcoming goal, we aim to verify the scalability of the system in a larger setup which also will allow us to test timeliness properties. From early tests we know that the individual dynamicity and automation features make fast sharing of updated metrics and quick reactions possible, but we still need to prove that the system can provide real-time data in realistic end-to-end scenarios. Finally, we plan to verify the practicality of the multi-tenancy features by running multiple parallel use-cases in an advanced test environment.

IV. Conclusions

We have outlined the properties required to monitor VNF instances in a 5G system, and have presented a software-defined (i.e., programmable) monitoring framework which fulfills these requirements. Our framework provides the means for flexible and programmable monitoring deployment, single-touch and dynamic monitoring configuration, and scalable, multi-tenant capable transport of monitoring results. Finally, we have discussed the identified shortcomings of the current framework and presented ongoing work in this respect, as well as future directions.

Acknowledgment: This work was in part supported by the Swedish VINNOVA project Kista 5G Transport Lab. The authors also want to thank Christofer Flinta and Catalin Meirosu for many constructive discussions and comments.

References

[1] J. F. Monserrat *et al.*, "Rethinking the mobile and wireless network architecture: The metis research into 5g," in *EuCNC*, June 2014.
[2] A. Sheoran *et al.*, "An empirical case for container-driven fine-grained vnf resource flexing," in *IEEE NFV-SDN*, 2016.
[3] S. Sharma and B. Coyne, "Devops for dummies," *2nd IBM limited edition*, 2013. [Online]. Available: http://www.ibm.com/ibm/devops/us/en/resources/dummiesbooks/
[4] A. Rostami *et al.*, "Multi-domain orchestration across ran and transport for 5g," in *ACM SIGCOMM'16 Demo Session*, 2016.
[5] D. Trihinas *et al.*, "JCatascopia: Monitoring elastically adaptive applications in the cloud," in *IEEE/ACM CCGrid*, 2014.
[6] P. Naik *et al.*, "NFVPerf: Online Performance Monitoring and Bottleneck Detection for NFV," in *IEEE NFV-SDN*, 2016.
[7] G. Gardikis *et al.*, "T-Nova deliverable D4.42: Monitoring and maintenance," 2016. [Online]. Available: http://www.t-nova.eu/results/
[8] G. Marchetto, R. Sisto, W. John *et al.*, "Final Service Provider DevOps concept and evaluation," *ArXiv e-prints*, vol. 1610.02387, 2016. [Online]. Available: http://arxiv.org/abs/1610.02387
[9] S. Van Rossem *et al.*, "NFV Service Dynamicity with a DevOps Approach: Insights from a Use-case Realization," in *IFIP/IEEE IM, Experience Session*, 2017.
[10] F. Moradi, C. Flinta, A. Johnsson, and C. Meirosu, "ConMon: an automated container based network performance monitoring system," in *IFIP/IEEE IM*, 2017.
[11] W. John *et al.*, "Scalable Software Defined Monitoring for Service Provider DevOps," in *EWSDN*. IEEE, 2015.