Scalable Monitoring for Multiple Virtualized Infrastructures for 5G Services

Panagiotis Trakadas[1], Panagiotis Karkazis[1], Helen-Catherine Leligou[1], Theodore Zahariadis[1],
Wouter Tavernier[2], Thomas Soenen[2], Steven van Rossem[2], Luis Miguel Contreras Murillo[3],
[1]Synelixis Solutions Ltd., Chalkida, Greece, {ptrak|pkarkazis|nleligou|zahariad}@synelixis.com}
[2] University of Ghent, Belgium, {wouter.tavernier|thomas.soenen|steven.vanrossem@ugent.be}
[3] Telefonica, Spain, {luismiguel.contrerasmurillo@telefonica.com}

***Abstract*— This paper presents the high level architecture and functionality details of the monitoring framework that has been implemented and integrated within the SONATA project, in order to support the management of 5G services under the SDN/NFV paradigm. The innovative framework, extending the functionality of Prometheus.io, is unique in its support for multiple types of virtualized infrastructure, supporting multiple Points of Presence (PoP), is extensible using Websockets, and is fully available in open-source.**

Keywords*— *NFV/SDN, Cloud Computing, Monitoring, Virtual Machines, Containers, Network Services

I. INTRODUCTION

In the next years, 5G infrastructure will become a ubiquitous, flexible, broadband and programmable network that will be in the core of every social, business, and cultural process, enabling both economic growth and social prosperity. In order to achieve this goal, the 5G vision poses significant technical challenges that must be fulfilled, including the concept of agile programmability and supporting the introduction of management mechanisms for the efficient instantiation of innovative services across heterogeneous network components, virtualized infrastructures and geographically dispersed cloud environments.

One of the important issues to be addressed in this new era of 5G service management is related to network and service monitoring, demanding for collecting data and metrics on the performance and usage of the resources involved in the lifecycle management of 5G services. However, the already available monitoring tools do not achieve the requirements stemming from the services envisioned in the 5G landscape, since they are in most of the cases: (i) intrusive and heavy-handed for short-lived, lightweight network function instances, (ii) not able to follow the fast pace of management changes enforced by continuous dynamic scheduling, provisioning and auto-scaling, (iii) not covering the requirements of all the involved emerging technologies, including deployments in both hypervisor-based and containerized manner, as well as monitoring data collection from different cloud environments (OpenStack, VMWare, etc).

This paper presents the monitoring framework that has been implemented within the SONATA European project, in compliance with the ETSI NFV MANO specifications, providing an interactive monitoring framework capable of offering real-time data collection, processing and alerting to all stakeholders of an SDN/NFV-enabled service platform, i.e. service developers, service platform operators and end-users, under heterogeneous cloud-enabled computing environments.

II. STATE OF THE ART

Network monitoring has been an active research topic for more than three decades. Well-established protocols such as SNMP [1] and Netflow/IPFIX [2] are already successfully applied for gathering network metrics through either passive or active measurements. However, network metrics in isolation are not very useful in services-oriented systems; they have to be aggregated and consolidated with service- and resource-related information to produce an integrated picture of the performance of the provided service. Hence, another category of monitoring tools is mostly focusing on computation, storage and memory resources of the infrastructure or the deployed service/application, such as Nagios [3] and Zabbix [4]. One of the most advanced and modern monitoring tools is Prometheus [5], that is an open-source service monitoring system, based on time series database that implements a highly dimensional data model, where time series are identified by a metric name and a set of key-value pairs. Moreover, Prometheus provides a flexible query language, allowing slicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts, while it is integrated with visualization tools (Grafana and PromDash). Most importantly, Prometheus provides probes that allow bridging of third-party data into Prometheus, including cAdvisor and collectd in a “pull” fashion, but also supports “push” through an already implemented gateway.

Recently, the concept of Software Defined Networking (SDN) and Network Function Virtualization (NFV) in combination with the advent of Cloud Computing and containerization of services, has dictated the implementation of monitoring tools in conformance with the respective technologies, that will allow the retrieval of SDN-based, per-flow information directly via the API of the Openflow controller (e.g. OpenDaylight Statistics REST API [6]), the collection of monitoring data within Docker containers via cAdvisor tool [7] as well as the performance monitoring of cloud infrastructures and instantiated services, such as Monasca for OpenStack [8]. Following these trends, the programmability of 5G software network infrastructure will require a flexible and expandable monitoring tool to complement the management of the deployed innovative services, integrating the benefits of the abovementioned tools in a unified framework. During the last years a remarkable effort has been made on the development and integration of such monitoring frameworks under different viewpoints: In [9], the authors introduce a management solution for cloud federation that automates service provisioning and achieves seamless deployment of services across a future internet cloud federation, while also monitoring the resources and data which can be aggregated with a common structure; however, this framework lacks inherent support for NFV deployment. In [10], the challenges of proper NFV monitoring are discussed, focusing on the process of collecting NFV Infrastructure (NFVI) metrics and processing them at Virtualized Infrastructure Management (VIM) level. Finally, in [11], the authors present a monitoring and discovery framework for self-organized network management in virtualized and software defined networks, that although relevant to the management of 5G services under the SDN/NFV paradigm, it is focusing on sensor networks.

In a pure programmable network environment, such as in 5G landscape, there are also monitoring metrics associated with the performance of the service platform itself, such as Orchestrator signaling throughput and response time, as well as other requirements, such as the ability for metrics and alerts modifications during runtime which are currently not covered by any of the current monitoring frameworks, leading to the need of implementing a monitoring framework with advanced capabilities, as described below.

III. Monitoring Framework Requirements for 5G Services

This section presents the requirements related to monitoring arising from the use case scenarios of SONATA EU-funded project [12], acting as the drivers for the monitoring architecture design that has been followed (see Table 1).

In particular, the fulfillment of VNF specific monitoring requirement demands the implementation of an HTTP API as well as a real-time mechanism that will allow developers/users to monitor performance data related to their deployed Network Services. Although not specifically mentioned in the above-mentioned requirements, it is also required that monitoring system must collect data from VNFs deployed on virtual machines and containers in different infrastructures. Additionally, in order to facilitate the resource orchestration process, SONATA monitoring system must collect and offer information related to the available resources of the infrastructure, as mandated by VNF Placement. For example, the network service developer must be informed whether special conditions required for the service deployment are satisfied in a particular NFVI or be able to modify orchestration parameters. Thus, monitoring system must be able to collect data from the underlying infrastructures comprising the SONATA ecosystem. The collection of information from the above-mentioned components will also address the requirement of VNF Status Monitor, providing service status information (e.g. error state). Apart from offering an API to developers for collecting and processing monitoring data related to their deployed NS/VNF, the monitoring system must be able to accommodate VNF-specific alerting rules for real-time notifications, as described in the Timely alarms for SLA violation and VNF Real-time Monitoring requirements. In this respect, the presented SONATA monitoring framework will offer the capability to developers to define service-specific rules, whose violation will inform them in real-time. Finally, there is one requirement related to the Quality of Service that demands special attention with regards to sampling period and monitoring accuracy and another one (Service Platform Scalability) directly related to scalability of the SONATA monitoring framework with respect to the Service Platform and respective infrastructures. Hence, the monitoring solution must comply with the scalability requirement dictated by 5G network infrastructure and services.

Table 1: Requirements fulfilled by the SONATA Monitoring Framework

Req. name	*Description*	*KPIs*
VNF specific monitoring	SONATA Service Platform shall expose service and VNF metrics to the network application.	Availability of an API for VNFs capturing monitoring metrics.
VNF status monitoring	SONATA should provide a high level state for each VNF, e.g., (i) deployment, (ii) operating, (iii) error.	Provide a dashboard displaying status data
VNF placement and metrics modification during runtime	The programmability framework shall allow the customer to deploy VNFs at arbitrary points into the network and modify metrics parameters in runtime.	SLA/QoS metrics related to deployment time, cost, etc as well as interfaces for timely modification of metrics and thresholds.
Timely alarms for SLA violation	The monitoring system must provide alarms for SLA violations in a timely manner.	Proven performance and scalability of the selected message bus and websocket creation
VNF real-time monitoring	VNFs will generate in real time information useful for monitoring and response.	Monitoring frequency, time to process alerts.
Quality of service monitoring	Metrics generation and degradation detection of network traffic, should be supported and reported.	Traffic QoS, packet loss, delays.
Monitoring Framework Scalability	The monitoring framework must be scalable to support multiple and heterogeneous infrastructures and a high traffic load.	Support for multi-PoP and multi-tenancy federated environment.

IV. High-Level Architecture and Functionality of Monitoring Framework

In a nutshell, the SONATA monitoring framework collects and processes data from several sources, providing the developer the ability to activate metrics and thresholds in order to capture generic or service-specific behaviour. Moreover, the developer can define rules based on metrics gathered from one or more VNFs deployed in one or more NFVIs in order to receive notifications in real time. In general, the developer is able to subscribe to a message queue or he can get the alert notifications by email and/or SMS on his smartphone. Most importantly, monitoring data and alerts are also accessible through an API or directly accessing a websocket URL. The internal architecture of Monitoring Framework is depicted in Figure 1 and explained in the next subsections.

A. *Collecting data from several sources*

It is of paramount importance to collect monitoring data from as many as possible sources. In the implemented framework, there are four different types of sources for collecting data: 1) *container probe* which runs inside the container-based VNFs to collect data related to their performance, 2) *VM probe* that collects data from Virtual Machines (VMs) hosting VNFs, 3) *OpenFlow probe* which is a Python software that utilizes OpenDayLight API to collect data from the OpenFlow controller, and 4) *OpenStack probe* that has also been developed as a software module (in Python language) that uses OpenStack API to collect data from all OpenStack components.

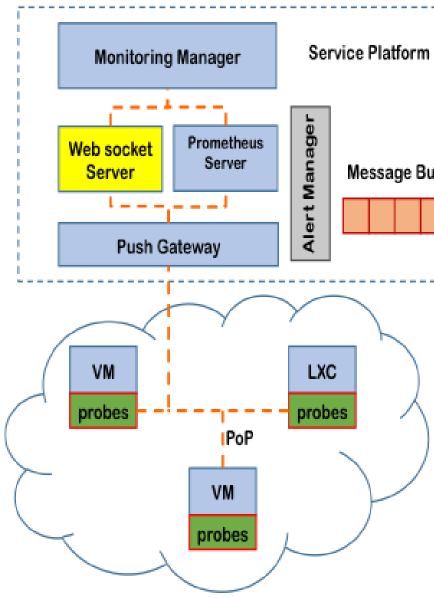

Figure 1: Monitoring Framework high-level architecture

B. *Push Gateway*

This component is part of the open source Prometheus monitoring solution [5] that has been adopted and extended to cover the needs of SONATA Monitoring Framework. Push Gateway is utilized so that the probes/sources use HTTP POST method to “push” monitoring data to the Push Gateway, while Prometheus server collects the data in a predefined time interval. The advantage of this approach is that in the case of the deployment of a new service, there is no need for the Prometheus monitoring server to search for data related to the newly deployed VNF, but rather collect them from the PushGateway.

C. *Prometheus Monitoring Server*

Prometheus is an open-source service monitoring system, based on time series database that implements a highly dimensional data model. A time series entry is identified by a metric name and a set of key-value pairs. Prometheus has a sophisticated local storage subsystem (LevelDB), which is essentially dealing with data on disk and relies on the disk caches of the operating system for optimal performance. Prometheus server is responsible for collecting the data and communicating with the time-series database for retrieving data upon request.

D. *Monitoring Manager*

Monitoring manager is a Django-based server that offers APIs to the users with respect to the monitoring data of their instantiated 5G services, including: 1) the relation among services, network functions, NFVIs and users, 2) the ability to modify rules and thresholds during service/function runtime, 3) the reconfiguration of Prometheus server, 4) the ability to define the notification methods in case of alert generation, 5) the definition of a new websocket to get data in real-time and many other features. The interested user can find information at http://sp.int3.sonata-nfv.eu:8000/docs/.

E. *Alert Manager*

As previously discussed, the Alert Manager is responsible (along with the implementation of a message queuing mechanism, such as RabbitMQ) for sending notifications about firing alerts to the subscribed users. After this notification, the user can take advantage of the API to further investigate the fault or activate a websocket to receive real-time monitoring data.

F. *Websocket server*

The implementation of websockets (Tornado web server) allows the user to collect streaming data from VNFs that have been deployed in the Service Platform. This is highly beneficial to the developers, as they would be able to monitor the performance of a new service in real environment. Prior to the establishment of a new websocket, the user must be aware of the metrics collected per VNF, the VNFs comprising his deployed Network Services and other related information and this information is already provided by the existing Monitoring Manager API framework, as depicted in Figure 2. After selecting the VNF and the respective metrics to be sent, the user requests the creation of a new websocket from the Monitoring Manager. After checking the validity of the request, the Monitoring Manager communicates with the Websocket server that creates and sends a new URL for the user to connect to and where metric values are pushed.

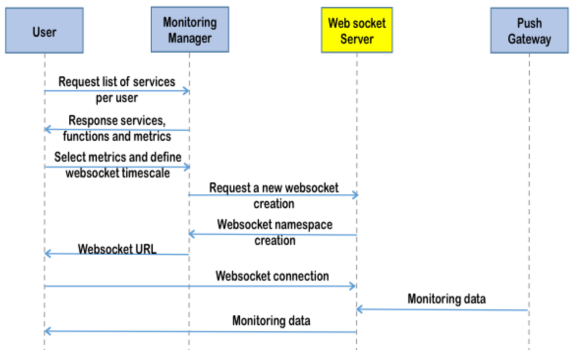
User
Monitoring Manager
Web socket Server
Push Gateway
Request list of services per user
Response services, functions and metrics
Select metrics and define websocket timescale
Request a new websocket creation
Websocket namespace creation
Websocket URL
Websocket connection
Monitoring data
Monitoring data

Figure 2: Websocket interactions.

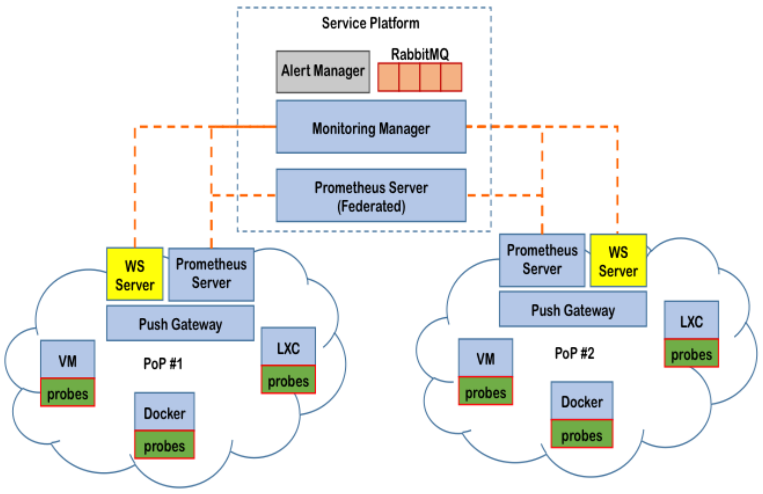

Figure 3: Architecture addressing scalability requirements with respect to the monitoring framework.

V. Scalability and Distributed Architecture

One of the cornerstones of the monitoring framework implementation was to deliver a carrier-grade solution that would fulfill scalability requirements in a multi-PoP environment. As can be noticed from Figure 3, several components of the Monitoring Framework had to be distributed across the SONATA Points of Presence (PoPs). First, each PoP must have its own websocket server to accommodate developers' demands for streaming data, although the management of websockets is handled by the Monitoring Manager instance in a centralized way. Second, Prometheus Monitoring servers follow a distributed (cascaded) architecture. The local Prometheus servers collect and store metric data from the VNFs deployed in the PoP, while only the alerts are sent to the federated Prometheus server for further processing and forwarding to the subscribed users. Moreover, the alerting rules and notifications are based on monitoring data collected in different PoPs and thus the decision must be made on a federation level. Another scalability requirement concerns the large flow of data from the monitoring probes to the Monitoring Server and its respective database that might affect the service performance in extreme cases. In this respect, an architectural decision to address this scalability issue was to support a distributed architecture regarding the monitoring server and its database, working in a cascaded fashion along with proper modifications on component level. In particular, the functionality of the monitoring probe will change so that it will not send data to the monitoring server in cases where the value difference is less than a threshold defined by the developer. The same will be the case in the communication between the monitoring server within a NFVI and the monitoring server in the Service Platform.

VI. Conclusions and Future Work

The innovative SONATA monitoring framework builds further on state-of-the-art technology like RabbitMQ, Prometheus and Websockets, enabling a multi-PoP framework with extensible and user-friendly monitoring of NFV services involving both containers and Virtual Machines, empowering service management components to dynamically react on triggered monitoring alerts. As a future work, in the context of 5G-TANGO EU-funded project (http://5gtango.eu/), the described Monitoring Framework will be further enhanced by introducing the concept of autonomic management, as described in the respective ETSI documents [13].

Acknowledgment

This work has been performed in the framework of the SONATA and 5GTANGO projects, funded by the European Commission through the Horizon 2020 and 5G-PPP programmes.

References

[1] J.D. Case, et al., RFC1157 Simple Network Management Protocol (SNMP), IETF, 1990

[2] B. Claise, Ed, RFC3954, Cisco Systems NetFlow Services Export Version 9, IETF, 2004

[3] Nagios monitoring solution, https://www.nagios.org/

[4] Zabbix, Enterprise class Open Source Network Monitoring, http://www.zabbix.com/.

[5] Prometheus open source monitoring solution, https://prometheus.io/

[6] OpenDaylight Statistics REST API, https://www.opendaylight.org/

[7] cAdvisor, Monitor containers, https://hub.docker.com/r/google/cadvisor/

[8] Monasca OpenStack project, https://wiki.openstack.org/wiki/Monasca

[9] Theodore Zahariadis, et al., "FI-Lab: Managing Resources and Services in a Cloud Federation supporting Future Internet Applications", 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014).

[10] G. Gardikis, et al., "An Integrating Framework for Efficient NFV Monitoring", Proceedings of the IEEE NetSoft Conference and Workshops, Seoul, Korea, 6-10 June 2016, pp. 1-5.

[11] A. L. V. Caraguay, L. J. G. Villalba, "Monitoring and Discovery for Self-Organized Network Management in Virtualized and Software Defined Networks", Sensors, 2017, 17, 731, DOI: 10.3390/s17040731.

[12] SONATA project, http://sonata-nfv.eu/

[13] ETSI GS AFI 002, v1.1.1, Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture, 2013.