Real-Time Management and Control of Monitoring Elements In Dynamic Cloud Network Systems

Francesco Tusa, Stuart Clayman, Alex Galis
Dept. of Electronic Engineering, University College London, London, UK
Email: francesco.tusa@ucl.ac.uk, s.clayman@ucl.ac.uk, a.galis@ucl.ac.uk

***Abstract*—This paper explores new scenarios where Cloud Network Service Providers take advantage of using more flexible resource management and orchestration solutions in the form of dynamic virtualised compute, network and storage resources. The main focus of this work is to analyse how those challenges will considerably impact the requirements of the monitoring process. A framework in the context of 5G is here presented to support the dynamic on-demand management, configuration and control of a monitoring subsystem which: can easily scale up / down according to the number of running entities in the system as result of the instantiation / termination of multiple services; can provide mechanisms to dynamically activate / deactivate its constituent elements on-demand according to the type of services to be monitored; and can provide mechanisms to dynamically adjust the configuration if its elements. Experimental outcomes, where a Monitoring Controller was used to adjust the measurement collection / sending rate of the probes in the monitoring subsystem on the-fly are also presented. The paper shows how this prevented the transmission of vast amounts of data when the number of virtual entities and related monitoring probes in the system scaled up to hundreds of elements.**

***Index Terms*—Cloud monitoring, resources monitoring, dynamic control, real-time control, on-demand reconfiguration, network service orchestration, NFV, SDN, 5G.**

I. INTRODUCTION

Cloud computing service providers have gradually been adopting new technologies to deliver services to their customers over Wide Area Networks. Network resources are now being abstracted, shared and delivered as a bundle to different tenants along with computational resources. In order to achieve that, different Cloud infrastructures can be interconnected via a combination of Software Defined Networks (SDN) and Network Function Virtualization (NFV) technologies implemented by using heterogeneous types of software and / or (virtualised) hardware entities.

Due to the intrinsic high-dynamic nature of the pattern of customers service requests in these scenarios, the required resource management and orchestration processes may conveniently rely on the dynamic instantiation of ephemeral virtual entities associated to different type of resources (e.g., computing, network and storage), which are likely to be created and destroyed during a relatively short period of time.

Even during their short lifetime, those virtual entities that have been instantiated for the implementation of those services, will certainly generate data that is worth observing and collecting. This monitoring process will not only allow measuring the performance of the services themselves, but will also provide a valuable source of information related to the overall behaviour of the system. The presence of a monitoring process in the system's control loop becomes vital to ensure that a proper feedback is continuously provided to the functions that take care of the services' life-cycle management, configuration, and orchestration as they are dynamically created and destroyed.

5G Network Service Providers have started reconsidering their own core infrastructures according to a new more flexible software-based approach – Software Defined Infrastructures, i.e., a combination of NFV, SDN and Cloud technologies designed to fulfil customers' demand via relatively complex software systems that perform the end-to-end orchestration of network services on heterogeneous resources, even across administrative borders. In this multi-technology, multi-domain scenario, logically interconnected and virtualised resources are dynamically allocated / deallocated on demand following the pattern of the network services instantiation requested by the customers. Services can include computational, storage, and network resources that can be allocated as Virtual Machines (VMs), Containers, virtual storage devices, and virtual links.

As already mentioned, the instantiation of those virtualised elements is very likely to be performed according to a highly dynamic pattern and can possibly scale up to reach hundreds or thousands of running entities. As a consequence, firstly the monitoring subsystem should be able to activate / deactivate the required monitoring entities according to this dynamic pattern and also to the specific type of services to be monitored (i.e., the type of virtual resources a service is formed of). Secondly, the monitoring subsystem should be able to cope with the potential high number of (virtual) entities that may need to be managed, orchestrated, controlled and monitored during the execution of the above services. The higher the number of entities that are up and running in the system, the greater the amount of monitoring data that will be collected and sent over the network.

For this reason, some of the parameters of the system may have to be adjusted at run-time according either to the actual number of running entities or to the occurrence of specific events. For example, as the number of entities grows, the frequency at which measurements are collected during the monitoring process may need to be adjusted to avoid potential network congestions. Conversely, if there is either a potential fault detected in the system, or an anomalous behaviour in one of the components of a service instance, finer grained monitoring data might be required and the rate at which measurements are collected may have to be increased.

This will introduce additional requirements for the whole resource infrastructure's management system. As such, this paper is focused on the monitoring process and describes the set of features a monitoring subsystem should support to comply with the requirements of a dynamic cloud network environment. Most of the current existing monitoring solutions are based on pre-configured static deployment strategies that are difficult to customise at run-time. Using such systems in scenarios that require dynamic adaptation and control of their elements (such as in a 5G environment) would not be appropriate and would make the implementation of a fully-dynamic software-defined infrastructure more difficult or even not feasible.

This paper highlights the new features that have recently been added to *Lattice*, a monitoring framework that was developed and already used to monitor virtual computational [1] and networking environments [2] environments in the context of the EU FP7 project RESERVOIR [11]. The new Lattice's functionalities fulfil most of the above mentioned requirements and allow dynamic activation, control and management of the main elements constituting a monitoring subsystem.

The paper is organised as follows. Section II presents the state-of-the-art on existing monitoring systems. Section III provides an overview of the Lattice Monitoring framework and describes how it was augmented to be used in the context of dynamic cloud network systems. Section IV describes an actual cloud network scenario related to 5G where Lattice was used for monitoring resources and services. Section V provide further details on how dynamic control can be achieved in that scenario via reconfiguring relevant probes on-demand. Finally Section VI draws some conclusions and foresees potential future works.

II. Related Works

This section presents a brief overview of the state-of-the-art on monitoring systems: it starts discussing traditional data centers and grid monitoring solutions and then moves to systems specifically designed to monitor Clouds environments.

The reader will see that, on the one hand, the former systems lack of flexibility and cannot easily used in scenarios where the topology of the entities to be monitored changes during relatively short time intervals; on the other hand, the latter ones, although designed to work on less static environments, still miss dynamic programmability features that allow changing the behaviour of the system at run time. The outcome of this analysis was used as a starting point to identify functional gaps and requirements for a desired Cloud Network monitoring system.

Nagios [7] is one of the best-known open source tools for monitoring IT infrastructures and adopts a centralised client/server architecture – a server is responsible of gathering monitoring information from remote monitored nodes for the sake of storage and visualisation. Nagios configuration is static and does not allow dynamic control of the services and entities to be monitored. Nagios supports the development of plug-ins to extend some if its functionalities, e.g., to provide a remote API for dynamically changing the system configuration. However, the supported API features are quite rudimentary and limited to the scope of the central monitoring server.

Zabbix [3] is an open source software that offers good performance for gathering data on large scale environments. It allows collecting monitoring measurements, statistics and performance data from servers, network devices, and applications. The system configuration can become complex when new entities or custom checks have to be added. Zabbix offers an API that can be used to retrieve and modify the configuration of the system in a programmatic way. This API is currently widely used to integrate Zabbix with third party software in order to automate routine tasks. Also in this case, the scope of the API is limited to the configuration of the central Zabbix monitoring server.

Ganglia [10] is another relevant system widely adopted to monitor grids and clusters of computers. It supports scalable monitoring by using a hierarchical distributed architecture relying on clients, agents and servers entities: an agent (gmond) sends resource usage information on demand to another agent (gmetad) that plays the role of higher level aggregator. Ganglia does not support any mechanisms for adjusting the configuration of the above agents dynamically and does not provide any remote APIs either.

Moving away from the traditional IT monitoring systems, Distributed Architecture for Resource manaGement and mOnitoring in cloudS (DARGOS) [9] is a completely distributed and highly efficient Cloud monitoring architecture to disseminate resource monitoring information. DARGOS ensures an accurate measurement of physical and virtual resources in the Cloud keeping at the same time a low overhead. In addition, DARGOS is flexible and adaptable and allows defining and monitoring new metrics easily. This solution has some commonalities with Lattice as it cares about optimising the network traffic by keeping a low overhead while the measurements are sent. However, DARGOS does not provide any control element nor an API for programming and automating either the deployment and (re)-configuration of the monitoring subsystem.

FlexACMS [4] is a framework to automate monitoring configuration related to cloud slices using multiple monitoring solutions. FlexACMS is able to detect new cloud slices created in the cloud platform and, for each new detected cloud slice, to trigger components that can configure the monitoring solutions, building the corresponding monitoring slice for that cloud slice. Although this solution represents a huge steps forward in respect to the automatic deployment and configuration of a monitoring subsystem, it still does not consider any on-demand adaptation e.g., according to the conditions of the system to be monitored, the number of running entities, etc.

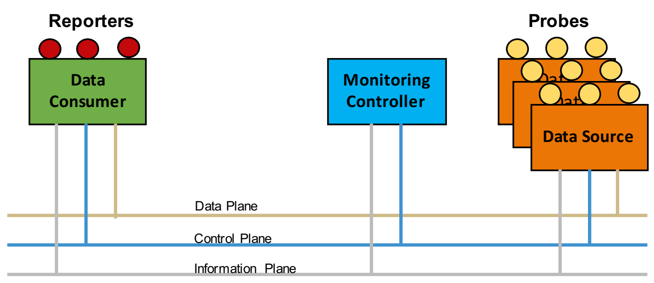


Figure 1: Lattice Framework Elements

III. Lattice Monitoring Framework

Lattice was initially designed and developed to be a highly customizable framework able to provide the software blocks that can be used to build an ad-hoc monitoring subsystem [2]. In the following, we will first provide an overview of the design and implementation of the main Lattice software elements; we will then discuss the requirements of a monitoring subsystem that would be beneficial to a highly dynamic cloud network scenario; we will finally describe how Lattice was augmented to support those requirements via the implementation of on-demand control, management, and configuration capabilities of its software elements.

A. Overview

Lattice is based on the concept of dynamically composable elements with data collection using *monitoring probes*, i.e., fully programmable Java software entities that can be used to retrieve measurements from different types of sources (see Figure 1). A probe usually collects measurements related to the elements of the system that needs to be monitored. The measurement process can be carried out either interacting with these elements directly (e.g., a physical host, a hypervisor, a network controller, an OVS switch, etc.) or with an existing higher level system that manages / controls / monitors those entities already (e.g., a Docker Engine, an OpenStack Ceilometer, an SDN Controller, etc.). In both cases, the probe will hide the required specific interaction mechanisms and will provide a uniform way to collect measurement data regardless of the specific type of entity to be monitored.

Probes are not the only elements required to build a customizable monitoring subsystem: the current specification of Lattice also provides *Data Sources*, *Data Consumers* and *Reporters*, where each of them implements specific functions required for the monitoring process (Figure 1, see [1] and [2] for further details). Although the first release of Lattice provided a good degree of flexibility in building the desired monitoring subsystem via deploying a customised topology based on the above software entities, an a-priori configuration of those blocks was still required, and an actual mechanism for configuring / controlling them at run-time was in the design but not yet implemented.

In order to develop a monitoring framework that was able to support the requirements imposed by highly dynamic environments such as the Cloud network infrastructures or the scenarios related to the emerging 5G, we extended Lattice to provide mechanisms enabling the dynamic control of the monitoring entities that are part of a Lattice-based monitoring subsystem. This allowed us to perform dynamic adaptation of the monitoring subsystem, according to the status of the main system that needs to be monitored (e.g., the status of resources and services, the overall number of instantiated entities, etc.) using both the Lattice Control and Information planes.

B. Requirements for a Monitoring Subsystem in dynamic Cloud network scenarios

A monitoring subsystem that supports the deployment and configuration of all the required monitoring entities at run-time, as well as their dynamic control would be desirable to have in the context discussed in the previous section. This would allow to dynamically reshape the deployment, configuration and behaviour of the monitoring subsystem according to the status of the system that needs to be monitored.

For doing that, the monitoring subsystem should (i) be able to collect and provide measurements from different types of resources, regardless of the technology used for their implementation; (ii) be able to easily scale up / down according to the number of running entities in the system as result of the instantiation / termination of multiple services; (iii) provide mechanisms to dynamically activate / deactivate its constituent elements (e.g., probes) on demand according to the type of services to be monitored; (iv) be able to provide mechanisms to dynamically adjust the configuration if its elements, e.g., the measurements collection / sending rate according to the status of the infrastructure, the number of running entities and the characteristics of the network service instances (e.g., defined thresholds and events, SLAs, etc.).

C. Dynamic control features

The Lattice Control and Information planes that were included as part of the design specification described in [1] and [2], have fully been implemented (using the ZeroMQ asynchronous messaging library [6]) in the latest version of Lattice.

The Information plane is used to share a topological view of the monitoring entities that are up and running in the monitoring subsystem (Probes and Data Sources as well as Data Consumers and Reporters), also including their properties / attributes and run-time status. The Control plane is utilised to enable one (or more) Monitoring Controller entities (represented in Figure 1) to interact with the other Lattice monitoring elements that are under their control in order to enforce control actions (e.g., activating / deactivating a given probe or a reporting mechanism, dynamically adjusting the measurements rate of a probe, etc.).

The Lattice Monitoring Controller has fully been designed, implemented and added to the other existing components of the Lattice framework. The Monitoring Controller supports the following dynamic run-time functionalities:

- to Start / Stop a Data Source monitoring agent on host / resource,
- to Load / Unload a probe (on a Data Source) with attributes (service id, NF id, Virtual Link id, etc.),
- to Activate / Deactivate a probe,
- to Set a probe measurements collection / sending rate,
- to Start / Stop a Data Consumer monitoring agent on host / resource,
- to Load / Unload a reporter (on a Data Consumer monitoring agent),
- to Configure a reporter to use a particular persistent storage system for storing the collected measurements.

These functions are exposed by the Monitoring Controller via a RESTful API that includes multiple separate endpoints. The API invocation on the Controller will first trigger the generation and then the transmission of proper control messages on the Lattice Control Plane. These messages implement a request-reply protocol involving the Monitoring Controller and the Monitoring elements that need to be controlled. More specifically, a light-weight RPC mechanism was implemented using the External Data Representation (XDR) – this allows sending proper messages to a monitoring entity that requires dynamic control / reconfiguration in order to remotely trigger the execution of proper methods / procedures. The result of each control operation is then passed back to the Monitoring Controller on the Control Plane and used as return value of the original REST API call.

These mechanisms will allow building the required monitoring control loop while enabling dynamic control features for its main monitoring elements. For example, probes can be loaded on specific Data Sources monitoring elements according to the "spatial" configuration of the services being instantiated. If the collected measurements highlight any abnormal behaviour of the system, the reporting rate of a specific probe can be increased in order to collect more accurate measurements and better identify any potential issues. On the other hand, when the number of monitored entities in the system has grown too much and the amount of measurements being sent / received on the Lattice Data Plane becomes so high as to degrade the performance of the main system (due to e.g., network congestion), the reporting rate of (some of) those probes can possibly be decreased in a dynamic fashion without negatively affecting the accuracy of the overall monitoring process.

Although Lattice can be used to build the monitoring functionalities of any kind of dynamic system requiring on-demand control, in the next section we will specifically consider a 5G multi-provider scenario where a Lattice-based monitoring subsystem was devised to support the dynamic deployment of network services. We will discuss and highlight how some of the Lattice features turned out to be pretty useful to support on-demand control and adaptation features for that system.

IV. Dynamic 5G Monitoring With Lattice

This section describes a multi-operator, multi-domain 5G scenario where Lattice was used to perform monitoring of resources and services. Network service instances formed of virtual computing elements (such as VMs, containers, etc.) as well as virtual links (e.g., flow entries, network tunnels, etc.) are created and destroyed dynamically according to the customers requests. Service components are orchestrated and deployed via the interaction of multi-domain MANO (Management and Orchestration) frameworks. Each MANO framework (namely Multi-domain Orchestrator – MdO) operates on a combination of heterogeneous resources belonging to Network Service Providers / Operators – where each resource partition is called a Resource Domain.

In a 5G ecosystem, different MANO frameworks belonging to different Providers that have stipulated specific business agreements, can interwork to create a globally federated infrastructure for the deployment of end-to-end network services. Figure 2 depicts this scenario and considers a single MANO instance operating over two internal Resource Domains. The representation is focused on the monitoring perspective and highlights two separate monitoring subsystem instances, both based on Lattice and deployed in the two Resource Domains.

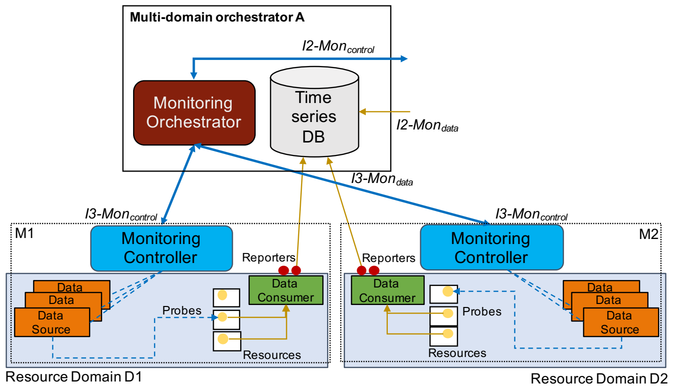


Figure 2: A 5G Service Provider using a Lattice based Monitoring subsystem

A specific component of the Multi-domain Orchestrator (MdO) – the Monitoring Orchestrator (in Figure 2, [8]) takes care of instantiating and configuring on-demand the required Lattice monitoring entities (i.e., Data Sources, Probes, Data Consumers, and Reporters) according to the services running on the infrastructure, and interacts with the Lattice Monitoring Controller via the I3-Mon (control) interface. The specific information to be monitored for each service is obtained from the QoS / QoE requirements specified by the customers, such as the SLOs (Service Level Objectives) within the SLA contract and / or rules describing the reaction of a service to particular events. Measurements collected by the probes are sent, via the Lattice Data plane, to the Data Consumers and then pushed to a permanent storage system (such as a time-series database) via the I3-Mon (data) interface.

In such a scenario, it will not be possible to make any a-priori assumptions on what kind of probes will be required to monitor a specific service until the latter will actually be up and running on the resource infrastructure. This is a consequence of how network services described by forwarding graphs are dynamically embedded on the available resources [5]. The dynamic deployment and configuration of different Lattice monitoring elements can resolve any potential issues related to that as it allows tracking the pattern of the services being deployed, and adapting the topology of the monitoring subsystem accordingly. Furthermore, as soon as the monitoring elements are put in place, it might also be required to slightly adjust their behaviour to deal with the additional dynamics affecting the status of the system. When one or more components of a service behave in an unexpected way, some parameters of the related monitoring probes could be reconfigured on-demand by sending control messages on the Lattice Control Plane to the related Data Sources, e.g., the measurement collection frequency can be increased to provide a more accurate monitoring information as feedback to the management and orchestration components of the system.

In a similar way, many monitorable objects may become up and running in the system at a given moment in time as consequence of the need of scaling up the number of allocated entities due to e.g., a workload peak. Under these circumstances, the increased number of measurements coming from the existing monitoring probes that are sent out on the Lattice Data Plane might be the cause of the transmission of very large amounts of data over the network. The amount of traffic generated by sending those measurements can be estimated on a Data Consumer: the rate at which data is flowing into a given Data Consumer is constantly internally measured and can also be retrieved via the Monitoring Controller (i.e., through its RESTful API). If that rate becomes greater than a particular defined threshold, the frequency at which the measurements are collected from the probes and sent to that Data Consumer can be adjusted on-the-fly by controlling the probes configuration. This will have the effect of reducing the network traffic associated to the transmission of the monitoring data.

When using Lattice, this process can be done dynamically without the need to restart any of the monitoring subsystem components – the configuration of the probes can easily be adapted by sending appropriate control messages on the Lattice Control Plane to the relevant Data Sources. A more detailed description of this dynamic adaptation mechanisms, based on an actual test case, is reported in the next section.

The above monitoring adaptation logic was developed as a plug-in of the Monitoring Orchestrator represented in Figure 2: an algorithm receives as input (i) the incoming measurements rate estimated on the Data Consumer and (ii) the measurement rate of each probe sending data to that Data Consumer. The algorithm determines how those rates have to be adjusted to keep the incoming overall measurements rate on that Data Consumer below a given threshold. When this condition is not fulfilled, the Monitoring Orchestrator triggers a control event on the Monitoring Controller: proper control messages are generated and sent over the Lattice Control Plane to the correct Data Sources in order to adjust the rate of the relevant probes on-the-fly. The scenario discussed in this section was implemented in the multi-provider ecosystem envisaged by the 5G-PPP 5GEx project [12], and will further be described in the next section.

V. Real-time dynamic control of monitoring probes measurement rate

This section describes the outcome of a test performed on a simplified version of the scenario represented in Figure 2 (i.e., a multi-domain infrastructure of a 5G Service Provider), where a single Resource Domain and a Multi-domain orchestrator instance were considered. The test case was devised to learn and discuss additional details on the dynamic adaptation mechanisms supported by a Lattice based monitoring subsystem.

The testbed used for this experiment consisted of two separate physical machines connected to the same LAN, and each hosting different software components. The first one (the Resource Domain) was used during the test to run instances of: a Docker engine, a Monitoring Controller and a Lattice Data Consumer (DC) as well as multiple Data Sources configured to send measurements over to the above DC. The second machine hosted an instance of the 5G multi-domain orchestrator software.

During the test, several monitoring probes were created and activated on those Data Sources on-the-fly, as result of multiple network service instantiation requests processed by the multi-domain orchestrator. Each probe was activated and configured to collect monitoring information about the resource KPIs of the Docker containers instantiated to implement the above service instances. Once activated, each probe was configured to send measurements over the Lattice Data Plane at a pre-set default rate.

This experiment was part of the large-scale evaluation test of the 5G Provider multi-domain orchestration system, which consisted in assessing the behaviour of the resource orchestration functions under exceptional high load conditions. Although this test case reflects real-world situations expected in the 5G environments, for practical reasons it was carried out emulating the 5G customers' requests via a purposely devised software tool.

The test case helped to learn how the overall network traffic related to the transmission of monitoring data could be measured, controlled, and possibly reduced when Lattice was used for building up the monitoring subsystem. This was done by estimating the cumulative measurements rate on the deployed DC and by dynamically adjusting the measurements collection / sending rate of individual probes in the subsystem.

The cumulative measurement rate estimated on the Data Consumer DC can analytically be described by the equation:

$$DC_{rate} = \sum_{i=1}^{N_{probes}} r_i \qquad (1)$$

Where N_{probes} is the number of probes sending measurements to the Data Consumer DC, and r_i is the rate of the i^{th} probe. In order to keep the cumulative rate DC_{rate} below a given threshold $DC_{rate_{max}}$, the constrain expressed by Eq. 2 will have to be imposed:

$$DC_{rate} \leq DC_{rate_{max}} \tag{2}$$

The same constraint can be also used to derive the following inequality:

$$\sum_{i=1}^{N_{probes}} r_i k < DC_{rate_{max}}$$

Where the original rates are considered for all the probes and a scaling factor k for those rates is also introduced. The scaling factor has to be taken into account when calculating the new rates that should be set on (all) the probes sending measurements to that DC – Lattice allows performing this type of reconfiguration dynamically without the need of restarting any of the software elements of the monitoring subsystem. The rate scaling factor k can be calculated by solving the equation:

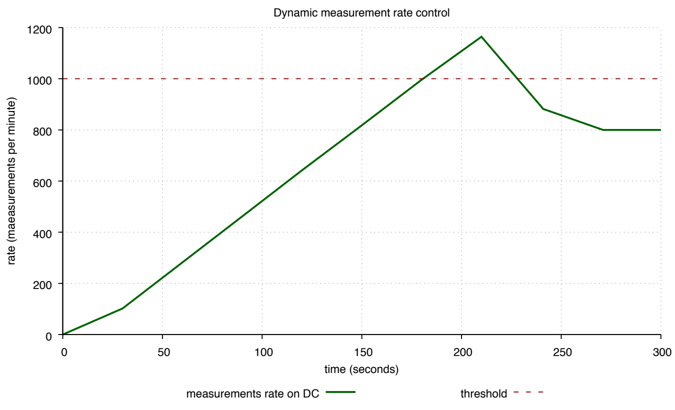
$$k \leq \frac{DC_{rate_{max}}}{\sum_{i=1}^{N_{probes}} r_i} \tag{3}$$

The Monitoring Orchestrator represented in Figure 2 was the component in the Orchestration system responsible to interact with the Lattice Monitoring Controller to retrieve the measurement rate estimated on the DC. It also implemented the adaptation logic described by Eq. 3 and was able to determine the right scaling factor k required to keep the measurements rate on that DC below the expected threshold (i.e., below $DC_{rate_{max}}$).

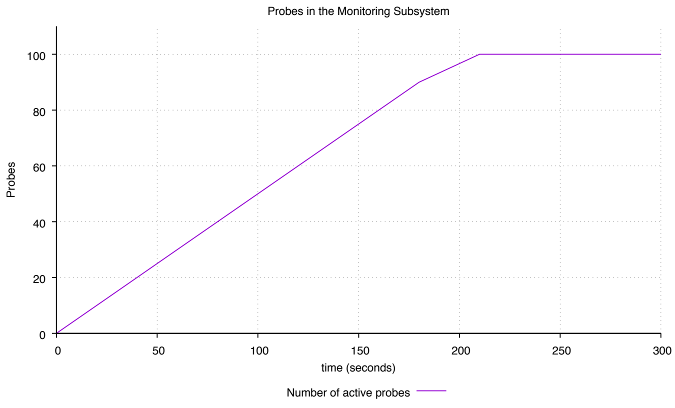
The measurement rate estimated on the DC (described by Eq. 1) was periodically retrieved via the Monitoring Controller API (every 30 s in this experiment) and the related collected values have been reported in Figure 3a. As the graph shows, the cumulative rate on the DC linearly increased along with the number of active probes in the system (which is depicted on Figure 3b). The higher the number of active probes in the system, the higher the cumulative rate of measurements received by the DC.

Figure 3a depicts how the dynamic adaptation of the monitoring probes' rate happened in this experimental set-up: when the DC cumulative rate value retrieved via the Monitoring Controller API became higher than $DC_{rate_{max}} = 1000$ measurements/min, the Monitoring Orchestrator reacted in order to adjust the configuration of the existing probes: a control action was propagated via the Lattice Monitoring Controller to the Data Sources where those probes were up and running. Control messages were sent over the Lattice Control Plane, and the rate of the probes was dynamically updated according to the scaling factor k calculated via Eq. 3. The reconfiguration happened without the need of restarting any of the components of the monitoring subsystem that were already up running.

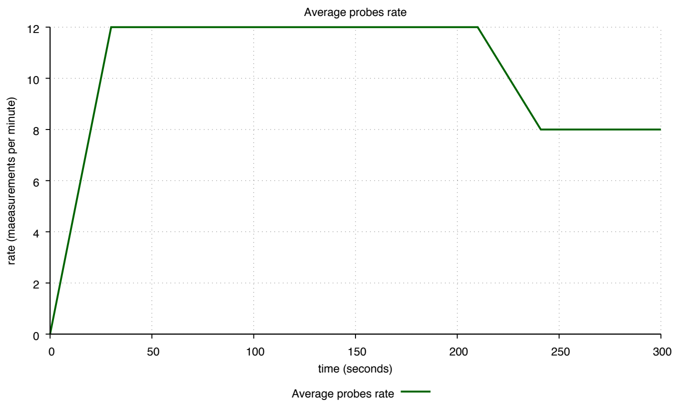
Figure 3b shows how the number of deployed probes in the system linearly grew during the test while the incoming service instantiation requests were processed (100 probes became up and running after 210 s). When a probe was created, it was also configured to report measurements at a default rate (expressed in measurements/min) to the DC – Figure 3c reports the average data rate (calculated as the mean value) for all the probes running in the system on the whole time window of the test.



(a) Cumulative rate on the Data Consumer



(b) Total number of running probes



(c) Average probes' measurement rate

Figure 3: Real-time control of the monitoring system

Figure 3c shows how the outcome of the above dynamic adaptation process affected the average measurement rate of the probes: as previously explained, when the overall measurement rate (expressed in measurements/min) estimated on the DC was greater than $DC_{rate} = 1000$, the individual rate of all the probes in the subsystem was reduced by the scaling factor k in order to keep DC_{rate} below the expected threshold.

As result, the average probes measurements sending rate was reduced from 12 measurements/min to 8 measurements/min – this had the effect to decrease the overall incoming measurements rate on the DC as well as to bring back the related estimated value below the expected threshold (Figure 3a, after about 220 s).

The main scope of this section was to provide a more detailed description of the real-time control mechanisms supported by Lattice. It highlighted, in particular, how an adaptation logic was implemented above the Monitoring Controller, and how automatic reactions could dynamically be triggered according to particular events detected in the monitoring subsystem in order to reconfigure some of its software elements on-the-fly.

The implementation of the Lattice control mechanisms is based on a simple request-reply protocol where packets are exchanged between the Monitoring Controller and e.g., a Data Source hosting a probe to be reconfigured. Thanks to the lightweight protocols and XDR serialisation methods used by the control system implementation, each control operation implies sending approx 10 KB of data over the control network. The communication overhead introduced by the above control mechanisms is objectively negligible compared to data that could flow from many different probes at a fixed (non-adjustable) rate.

Although the considered test case was specifically focused on controlling the overall amount of network traffic related to the transmission of monitoring data, alternatively, different types of events might have been used for triggering dynamic on-the-fly reconfiguration of the monitoring subsystem. For instance, the measurements sending rate of the probes may have been adjusted according to the values reported by the probes themselves in order to collect finer / coarser grained measurements in case those values became higher / lower than expected thresholds (e.g., related to the SLA objectives, service life-cycle events, etc.). Although these mechanisms are fully supported by Lattice, they have not been discussed here for the sake of space and might be the subject of future works.

VI. Conclusions and Future works

This paper presented and analysed the requirements that a monitoring subsystem should satisfy and the features it should support in order to effectively be used in highly dynamic Cloud network scenarios when hundreds of virtual entities are continuously created / destroyed by the resources management and orchestration system in a relatively short interval of time. It was discussed that the traditional approach to resource monitoring may no longer be effective. The monitoring subsystem should provide mechanisms to deploy, configure and control its entities on-demand and in real-time in order to properly monitor the service instances deployed on the infrastructure, and also properly react to the potential different events happening in the management and orchestration system.

In this paper, a 5G scenario where the Lattice framework was employed to implement a fully working monitoring subsystem was presented, and an actual test case demonstrated how Lattice was allowed to dynamically adjust the rate at which measurements are collected by the monitoring probes and sent over the network. This enabled controlling the volume of the total monitoring network traffic in the system and preventing potential congestions when the number of active probes in the system scaled up to reach hundreds of elements.

Future work will explore how Lattice can be used to support alternative test cases aimed at identifying what additional features would be desirable to integrate in the framework implementation. Exploring the performance of the Lattice control system might also be of interest in order to evaluate how the monitoring subsystem behaves both under stress and extreme work load conditions requiring the deployment of thousands of monitoring probes.

Acknowledgements

This work was partially supported by the EU projects: 5GEx – "5G Multi-Domain Exchange" (671636), NECOS – "Novel Enablers for Cloud Slicing" (777067).

References

[1] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L.M. Vaquero, K. Nagin, and B. Rochwerger. Monitoring Service Clouds in the Future Internet. In *Towards the Future Internet - Emerging Trends from European Research*. IOS Press, 2010.

[2] S. Clayman, A. Galis, and L. Mamatas. Monitoring virtual networks with Lattice. In *2010 IEEE/IFIP Network Operations and Management Symposium Workshops*, pages 239–246, April 2010.

[3] Zabbix Company. Zabbix - Enterprise open source monitoring software for networks and applications, 2018 (accessed on 2018-04-25). URL: https://www.zabbix.com.

[4] M. B. de Carvalho, R. P. Esteves, G. da Cunha Rodrigues, C. C. Marquezan, L. Z. Granville, and L. M. R. Tarouco. Efficient configuration of monitoring slices for cloud platform administrators. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, June 2014.

[5] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4):1888–1906, Fourth 2013.

[6] iMatix Corporation. Zero MQ Distributed Messaging, 2018 (accessed on 2018-04-27). URL: http://zeromq.org.

[7] Nagios Enterprises LLC. Nagios - The industry standard in IT infrastructure monitoring, 2018 (accessed on 2018-04-25). URL: https://www.nagios.org.

[8] W. Y. Poe, I. Vaishnavi, F. Tusa, J. Melián, and A. Ramos. System architecture of Intelligent Monitoring in multi-domain orchestration. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–5, June 2017.

[9] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini. DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds. *Future Generation Computer Systems*, 29(8):2041 – 2056, 2013. Including Special sections: Advanced Cloud Monitoring Systems & The fourth IEEE International Conference on e-Science 2011 — e-Science Applications and Tools & Cluster, Grid, and Cloud Computing.

[10] The Ganglia Project. Ganglia Monitoring service, 2018 (accessed on 2018-04-26). URL: http://ganglia.info.

[11] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan. The Reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):4:1–4:11, July 2009.

[12] A. Sgambelluri, F. Tusa, M. Gharbaoui, E. Maini, and L. Toka et al. Orchestration of Network Services across multiple operators: The 5G Exchange prototype. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–5, June 2017.