Towards Efficient and Adaptable Monitoring of Softwarized Mobile Networks

Towards Efficient and Adaptable Monitoring of Softwarized Mobile Networks

Alan Plascinskas, Xenofon Foukas, Mahesh K. Marina
The University of Edinburgh

***Abstract*—We consider the problem of monitoring in the context of emerging and future mobile networks which are shaping up to feature diverse set of services composed of customized chains of virtual network functions (VNFs) realized over (edge) cloud environments. In such a setting, not only is monitoring a critical component for service quality assurance, but it also needs to be efficient, adaptable and flexible. Informed by the experience analyzing state-of-the-art management and orchestration (MANO) platforms and monitoring solutions for softwarized mobile networks, we present our monitoring system design termed PliMon that aims to meet the above requirements by exploiting diverse temporal variability characteristics across different metrics (measurement features) and VNFs, and by grouping such metrics into tiers based on their relative significance. Using an experimental testbed, we verify the hypothesis that different measurement features and VNFs exhibit diversity in their variability and crucially show substantial reduction in monitoring overhead compared to representative monitoring solution from the literature. Additionally, we integrate PliMon with OSM, a well known open source MANO platform, and demonstrate the salient aspects of our approach using the integrated PliMon-OSM system.**

I. Introduction

As we head towards 5G, the mobile network architecture is undergoing significant transformation driven by the need to be flexible, programmable, cost-efficient and support a diverse array of services beyond traditional mobile broadband. This is evident in the way SDN and NFV principles (e.g., control-user plane separation, decoupling network functions from proprietary hardware into virtual network functions (VNFs)) are shaping the 5G architecture [1], [2]. These in turn form the basis of network slicing [3], a concept embraced in 5G to realize a multi-service infrastructure. The idea behind network slicing is to carve out a logical network instance for each service (a slice) over the shared virtualized infrastructure that is composed and customized with a set of VNFs as per the service requirements. The result is a network architecture comprised of infrastructure, network function and service layers along with a cross-cutting management & orchestration (MANO) entity.

To realize the multi-service vision over such a multi-layered architecture, service quality assurance is vital and a significant challenge. A key component of addressing this challenge and the focus of this paper is on monitoring the quality experienced by each service, including the underlying VNFs and the infrastructure utilization. Such monitoring can provide insights into the performance bottlenecks causing service guarantee violations, which in turn can guide the dynamic control of resource provisioning to avoid them. Clearly, a monitoring system needs to incur low overhead for tracking various metrics (measurement features) across VNFs. Equally, monitoring needs to be agile and responsive to varying conditions for timely detection of service performance bottlenecks to accordingly adapt resource provisioning along the service path.

In view of the above, we examine and analyze the state-of-the-art MANO frameworks and monitoring solutions as relevant to emerging 5G networks, both conceptually and empirically. This experience led us to our proposed monitoring system design, termed PliMon, that is pliable in the sense that it is adaptable to time-varying conditions across different VNFs and features while being efficient and flexible (§III). PliMon is built on two key ideas. First is the hypothesis that different VNFs and features experience different time-varying behaviors, which can be exploited to individually tailor the monitoring frequencies of each feature in such a way that monitoring overhead can be significantly reduced with no loss in effectiveness. We realize this idea in PliMon through a time series inference based method that tracks the variability of each feature to dynamically adapt its monitoring frequency. Second is the observation that only a subset of features need to be monitored under normal conditions and monitoring of other features can be activated on-demand. For example, tracking end-to-end service quality is sufficient normally and only when it shows a deterioration beyond a threshold, other features need to be monitored to gain a better insight on the root cause of the performance degradation. PliMon embeds this observation in its design through the notion of tiering by separating various relevant features of a VNF (or a set of VNFs) into different tiers (subgroups) ordered by importance with the lowest tier (0) consisting of most important and inexpensive features that need to be tracked continually and so on.

PliMon equipped with the above two ideas constitutes an advancement over the state-of-the-art on monitoring softwarized mobile networks. To drive home this point, we experimentally compare our PliMon approach with z-TORCH [4], a representative approach from the literature, and show that our approach leads to more than 40% reduction in monitoring overhead (§IV-B). Moreover, we go a step further and integrate PliMon into OSM [5], a prominent open-source MANO platform, thereby substantially expanding the latter's monitoring capabilities. We present two case studies of the integrated PliMon-OSM system that demonstrate the usefulness of our proposed approach in feature specific monitoring frequency adaptation and the value of multi-tier adaptive monitoring (§IV-C).

978-1-7281-4973-820$31.00 © 2020 IEEE

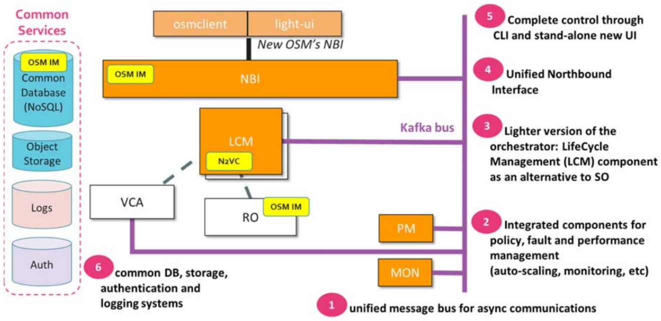
Common Services
OSM IM
Common Database (NoSQL)
Object Storage
Logs
Auth
osmclient
light-ui
New OSM's NBI
OSM IM
NBI
LCM
N2VC
VCA
RO
OSM IM
Kafka bus
PM
MON
5 Complete control through CLI and stand-alone new UI
4 Unified Northbound Interface
3 Lighter version of the orchestrator: LifeCycle Management (LCM) component as an alternative to SO
2 Integrated components for policy, fault and performance management (auto-scaling, monitoring, etc)
6 common DB, storage, authentication and logging systems
1 unified message bus for async communications

Figure 1. OSM Architecture [7].

II. Background and Related Work

Management and orchestration (MANO) platforms play a crucial role in the NFV lifecycle, as they are responsible for orchestrating the process of deploying, scaling and tearing down Network Services (NS) and their constituent VNFs by interfacing with underlying Virtualized Infrastructure Mangers (VIM) (e.g., OpenStack, AWS). Two of the most widely used MANO platforms are Open-Source MANO (OSM) [5] developed by ETSI; and ONAP [6] that is a Linux Foundation project formed through the merging of AT&T's ECOMP project and China Mobile's Open-Orchestrator project. Both OSM and ONAP follow a modular Docker-based micro-service architecture with different logical entities of the system – responsible for operations like the lifecycle management of VNFs, logging, authentication etc. – deployed in separate containers. As a reference, Figure 1 illustrates the architecture of OSM and its most significant modules, including Lifecycle Management (LCM), Performance Management (PM) and Monitoring (MON), which communicate using an Apache Kafka bus.

On the aspect of monitoring, the focus of this work, there is no industry-wide gold-standard. However, a plethora of practical monitoring systems already exist, either tailored for specific environments (e.g. Microsoft's Azure Monitor [8] for the Azure cloud) or targeting more generic infrastructure deployments, like Zabbix [9] and Nagios [10]. Such systems allow the monitoring of KPIs like CPU/network/memory/disk utilization, network flow information etc, both in terms of the underlying physical infrastructure and of the deployed virtual network functions and support the triggering of events and alarms whenever certain important KPIs cross some threshold.

Similarly, a number of proprietary monitoring solutions also exist in the mobile domain, with some noteable examples being Nokia's Wireless Network Guardian [11] and Amdocs' Deep Network Analytics [12]. In contrast to the cloud and data center monitoring solutions, these systems focus on analyzing data from the mobile networking domain, such as RAN related information (e.g. signal quality, missed scheduling deadlines, retransmissions, etc.) or information from the mobile core (e.g. bearer statistics and traffic load) in order to identify potential bottlenecks. The type of bottlenecks that are being considered in this case are different in nature and revolve around issues like interference, sudden traffic surges etc.

OSM and ONAP also have monitoring subsystems as part of their designs. In the case of ONAP, this is enabled through the Data Collection, Analytics and Events (DCAE) subsystem, which allows the gathering of any type of monitoring data and events from VNFs, physical hosts and the network through the use of a common event data model called VNF Event Stream (VES). OSM provides a multi-layered monitoring subsystem called MON, which can obtain information from various domains, by exploiting existing underlying monitoring solutions (e.g. Ceilometer for an OpenStack cloud, Juju Charms for the deployed VNFs and SDN for the network switches).

A key limitation of the aforementioned solutions is that, while they provide domain-specific or holistic monitoring frameworks, they do not focus on the monitoring overhead, which in the case of large scale NFV deployments for operational mobile networks can be significant. On the other hand, while a number of research works focus on solutions that can make monitoring as efficient and as fine grained as possible [13]–[15], they only focus on the reduction of the monitoring overhead and do not provide holistic monitoring frameworks. For example, [13] and [14] only consider monitoring the network flows and [15] is primarily concerned with the least intrusive way to obtain the generic compute metrics from cloud platforms.

z-TORCH [4], which can be viewed as a state-of-the-art monitoring system, is specifically designed for NFV deployments and is concerned with optimizing the quality of decisions (QoD) of MANO systems while minimizing the monitoring overhead incurred. z-TORCH employs various machine learning techniques like Q-learning and k-means clustering to adapt the frequency of monitoring to help make decisions on VNF placement across compute nodes. *One key limitation of z-TORCH is it uses the same monitoring frequency across all the monitored features and VNFs.* As a result, even when a small subset of VNFs (or a small subset of the VNF related features like CPU load and memory) present a behavior that fluctuates over time, all the VNFs and features in the system need to be monitored at the high frequency required to track the subset, thereby resulting in very high communication overheads. *Moreover, z-TORCH assumes that all the VNFs have an identical set of features to be monitored.* However, in 5G deployments, different VNFs could have a different set of features that must be monitored and therefore more flexibility is required in terms of the features to be monitored for each VNF.

III. PliMon

A key hypothesis underlying our proposed approach with PliMon is that different metrics (features) of different VNFs may have significantly different variability over time. This suggests that having different monitoring frequencies for different metrics and VNFs can lead to reduced monitoring overhead.

Figure 2 shows the schematic of PliMon's system architecture with the Monitoring Node (MNode) as a key component. It obtains measurements of a predefined set of metrics from each VNF via the measurement retrieval system (described later in this section). MNode feeds the obtained measurements,

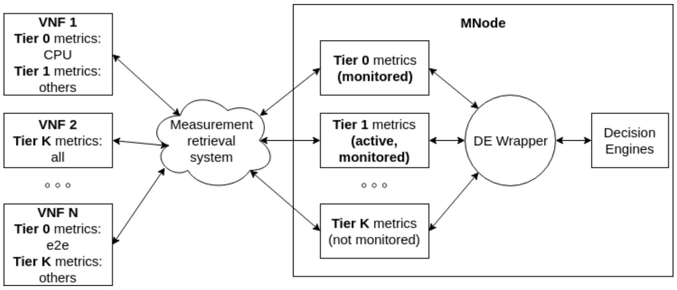

Figure 2. High level system architecture of PliMon.

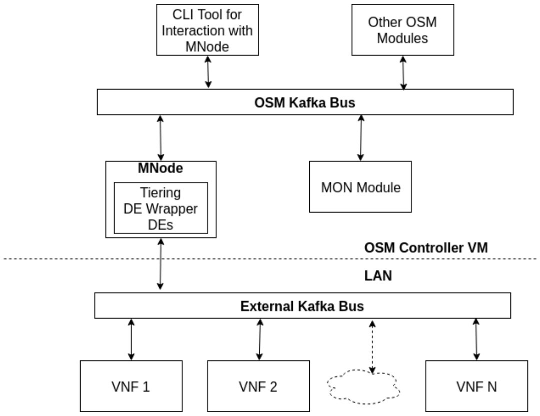

Figure 3. PliMon-OSM integrated system architecture.

through a DE wrapper, into the Decision Engine (DE), which returns the decision on the monitoring frequency of each VNF metric. The metrics are divided into tiers based on relevance, with the most important being in Tier 0. The measurements are performed only on the metrics that are in the active tiers. Note there are no restrictions on metrics as long as they can be expressed numerically and their past values carry some predictive meaning for the future values. This effectively makes PliMon a very flexible system that can be easily configured to monitor any set of metrics on any type of VNF.

A. Decision Engine

Here we describe the DE entity that adaptively decides the monitoring period (from among a discrete set of periods) for each metric in the active tiers.

We use a DE Wrapper entity to keep track of all the different VNFs, their metrics as well as their tiers. Thresholds (lower, upper) are specified for each metric where lower and upper thresholds indicate the crossing of the *safe* zone into the *danger* zone, which we want to be reported. More formally, when both thresholds are provided, the *safe* interval is $[lower, upper]$ and the *danger* interval is $(-\infty, lower) \cup (upper, +\infty)$. Intervals with just one threshold specified (e.g., upper bound) can be similarly defined.

In the rest of this subsection, we describe the DE operation for an arbitrarily chosen metric. DE decides on the monitoring period for the metric in question using time series based inference of Prediction Interval (PI) that we expect the future data to lie in and then picks the largest monitoring period in which we do not expect the PI to be violated with the given confidence. When increasing the monitoring period, we only allow increasing to the next larger period of the current one.

In order to track the temporal evolution of the metric, Y, we employ the well known Holt's double exponential smoothing method [16]. Firstly, we consider the Exponentially Weighted Moving Average (MA):

$$MA_t = (1-\alpha)MA_{t-1} + \alpha Y_t \tag{1}$$

and the Exponentially Weighted Moving Variance (MV) [17]:

$$MV_t = (1-\alpha)MV_{t-1} + \alpha(Y_t - MA_t)(Y_t - MA_{t-1}) \tag{2}$$

Next observation, Y_{t+1}, at time $t+1$ given the level and trend at t can be expressed as:

$$Y_{t+1} = l_t + b_t + \epsilon_t \tag{3}$$

Smoothed level at time t:

$$l_t = l_{t-1} + b_{t-1} + \alpha\epsilon_t \tag{4}$$

Smoothed trend at time t:

$$b_t = b_{t-1} + \alpha\beta\epsilon_t \tag{5}$$

where α and β are constant hyperparameters, which should be set by the administrator or via an additional fitting step (e.g.: using LSE), and ϵ_t is a zero-mean normally distributed error term with a variance for our purposes estimated to be MV_t.

From this we use the result from [16] to obtain a forecast h periods into the future:

$$\hat{Y}_{t+h} = l_t + hb_t \tag{6}$$

and its variance:

$$v_{t+h} = MV_t * (1 + \alpha^2 \sum_{j=1}^{h-1} (1 + j\beta)^2) \tag{7}$$

Firstly, given the observed data up to time t, we define a Prediction Interval (PI) as $[L, U]$ for a level of confidence Q such that the following holds:

$$Pr(L < p_{t+h} < U) = Q \tag{8}$$

We can easily calculate a prediction interval for a given degree of confidence. For example, PI for time $t+h$ at the confidence level of 95% can be expressed as:

$$\mu_{t+h} \pm 1.96\sqrt{v_{t+h}}$$

We can then pick the largest monitoring period $m \in MonPeriods$ such that $Pr(L < p_{t+m} < U) \geq Q$ holds, where L and U are lower and upper thresholds of the *safe* zone.

B. Tiered Monitoring

Tiers define the relevance that each metric carries with the most important metrics being in Tier 0. If the currently active tier is set to ξ, we will only be monitoring the metrics in tiers of equal or greater importance or, more formally, metrics in tiers $i \leq \xi, i \in Tiers$. Whenever we experience a *violation* in the tier ξ, we increase the active tier to $\xi + 1$. If we no longer have violations in tier ξ and $\xi - 1$ (the currently active tier and the tier before), we decrease the active tier to $\xi - 1$. The reasoning here is that we only want to decrease the tier when we no longer have violations in the currently active tier and in the tier that forced the active tier increase in the first place.

	HSS	**MME**	**SPGW**	**eNodeB**	**HSS**	**MME**	**SPGW**	**eNodeB**	**HSS**	**MME**	**SPGW**	**eNodeB**
CPU Utilization [%]												
Mean (μ)	0.39	0.42	0.42	16.48	0.36	0.43	0.39	16.93	0.37	0.43	0.38	16.33
Std (σ)	0.29	0.25	0.20	2.27	0.19	0.20	0.21	4.53	0.33	0.32	0.29	5.27
Memory Utilization [%]												
Mean (μ)	13.20	10.40	12.70	18.70	13.20	10.50	12.80	18.91	13.20	10.42	12.72	18.80
Std (σ)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.04	0.04	0.00
Network Utilization [%]												
Mean (μ)	0.01	0.01	4.12	11.18	0.01	0.01	2.96	8.92	0.01	0.01	0.01	3.32
Std (σ)	0.00	0.00	3.52	6.44	0.00	0.00	3.99	7.15	0.00	0.00	0.00	0.23

Table I
Results of experiments 1, 2 and 3 (from left to right)

C. PliMon Integration with OSM

To illustrate the possible avenues of practical application of the system within MANO frameworks, we chose to demonstrate how the system can be integrated into OSM. OSM was chosen as a representative example of a MANO framework suitable for research purposes due to its ease of use, ample documentation and low footprint [18]. Figure 3 illustrates how PliMon integrates with OSM. MNode is deployed as a Docker container within the OSM controller VM and connects to two Kafka messaging buses; an internal Kafka messaging bus of OSM to which it can export metrics upon request from a CLI tool or another module and an external Kafka bus that is used by MNode to produce monitoring frequency adjustment requests. The VNFs are configured to connect to the external bus and subscribe to the MNode requests as well as produce the metrics at the required frequency. The choice of using a second external bus for the monitoring related messages instead of the default OSM Kafka bus was made for added security, since allowing the VNFs to connect directly to the internal bus would provide them visibility to the OSM admin traffic.

IV. Evaluation

A. Verification of Metric Variability Hypothesis

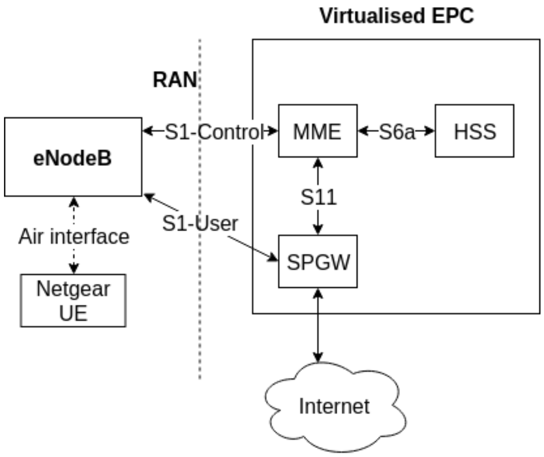

Figure 4. Experimental testbed setup illustrated.

Here, we verify that the variability of different metrics over the same VNF and the same metrics across different VNFs is not the same by setting up a series of end-to-end experiments on an OpenStack-based LTE testbed we deployed and illustrated in Figure 4. For the mobile core part of our testbed (HSS, MME and SPGW) we used the open source openair-cn [19] EPC functions, which we deployed in the form of OpenStack VMs, each with 2 virtual CPUs and 4GB of RAM. For the eNodeB, we used the OpenAirInterface open source LTE platform [20], which we deployed over a physical machine with an Intel i7-4770R CPU and 8GB of RAM.

Using our testbed, we performed 3 experiments using 1 or more UEs and measured the mean and standard deviation of the CPU, memory and network utilization of all the network functions. Our experiments were: 1) a speedtest using a single UE; 2) single UE video streaming at 1080p; and 3) a connection of 2 UEs in idle mode. The results of our measurements are summarized in Table I. In the case of experiments 1 and 2, we can observe very large standard deviations in the network utilization for the eNodeB and the SPGW, and almost constant memory utilizations. This observation suggests that network utilization might be a more indicative parameter of the service conditions for the eNodeB and the SPGW than the memory utilization. On the other hand, we can also observe that in the case of the HSS and MME, the network utilization presents hardly any variability. Similar observations can also be made for the case of experiment 3. Based on this discussion, we can conclude that different metrics can indeed present different variability behaviors and therefore such differences should be taken into account when monitoring the performance of network services.

B. Comparison with z-TORCH

We now evaluate the performance of the *varied* monitoring frequency approach of PliMon to the *non-varied* of z-TORCH [4]. To achieve this, we focus on two cases: one considering 3-feature VNF profiles (e.g. CPU, RAM and network) and the other considering 1200-feature VNFs (1200 arbitrary metrics). The 1200-feature VNF case reflects a large monitoring load, corresponding to scenarios where we want to monitor various fine-grained characteristics of each VNF, such as the load on each core of a multi-core CPU, fine grained network usage statistics for each network interface, etc. For the 3-features case we set their standard deviations (std) to be [0.01, 0.1, 0.3], while for the 1200-features case we replicated the 3 aforementioned stds 400 times ($3 \times 400 = 1200$ stds in total).

Both the *varied* and *non-varied* methods were compared in terms of the monitoring load required by the system and the decisions on creating the affinity groups, which are key to the z-TORCH VNF placement step. More formally, we define the monitoring load (in bytes) on the system as follows:

$$MLoad = \sum_{p \in M} \sum_{i \in metrics} \delta_{p,i} \theta_i |VNFs| \tag{9}$$

where M is a set of all the monitoring periods for a given run, $\delta_{p,i} \in \{0, 1\}$ is an indicator value showing if the data metric i was consumed in a given monitoring period p and θ_i is the size of data required to observe metric i in a single period. In our tests, monitoring data for all metrics is a single 64-bit floating point number, so $\theta_i = 8$B. For constant monitoring frequency among the metrics the above expression becomes:

$$MLoad_{non\_varied} = \theta |metrics||VNFs||M|. \tag{10}$$

After calculating the monitoring load which is the cost in our setup, we evaluate the qualitative difference in two approaches by comparing how the affinity binding decisions differ. More formally, we calculate the following metric:

$$Similarity\_slope = OLS(NV, V), \tag{11}$$

where OLS is an ordinary least squares estimator, taking the non-varied frequency affinity groups time series averaged over all tests NV and varied frequency affinity groups time series V at the same time points as NV and returns the least-squares unbiased estimator of the slope and intercept. A slope close to 1 would indicate that on average the two approaches yield similar results.

We evaluate the aforementioned metrics by running 20 tests for both the *varied* and *non-varied* approaches. The results in terms of the monitoring load are listed in table II, where we see how the varied metric frequency method saves 42% and 51% of monitoring load in the 3-feature and the 1200-feature cases, respectively, compared to the non-varied one. We then check that the affinity grouping decisions taken by the varied approach are comparable to the non-varied one. To verify this formally, we perform the least squares fitting as shown in the Figure 5, which yields a line with $Similarity\_slope = 1.05$ and $Similarity\_slope = 1.08$ for the 3-feature and the 1200-feature cases respectively. The similarity slope is compared to the identity line with slope 1. We can see that the fitted line is very close to that of the perfect fit and the data follows a linear model well, to verify this further, we calculate the coefficient of determination (R^2) [21]:

$$R^2_{V,NV} = 1 - \frac{Var(V - \hat{V})}{Var(V)} \tag{12}$$

where $\hat{V}$ indicates the perfect correspondence values from V to NV, i.e. prediction of V being equal to NV at a point in time. The R^2 given here is interpreted as $1 - FVU$, where FVU is a fraction of variance that is unexplained. Calculating the value from the test data gives: $R^2_{V,NV} = 0.989$ and $R^2_{V,NV} = 0.999$ for the 3-feature and the 1200-feature cases respectively. A value of 0 would mean that no covariance is explained by the model while a value of 1 means that the model captures the covariance perfectly. The values we obtained indicate that the identity line captures the relationship between the values very well.

Mode	3-feature VNFs	1200-feature VNFs
Non-varied	9.5MB	4.368GB
Varied	5.5MB	2.141GB
Load savings	**42%**	**51%**

Table II
MONITORING LOAD COMPARISON BETWEEN PLIMON (VARIED) AND Z-TORCH (NON-VARIED)

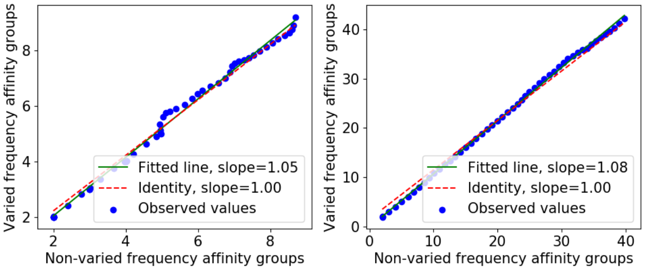

Figure 5. z-TORCH affinity groups for VNF placement with varied and non-varied monitoring frequencies at the same time points.

C. Case Studies

Here we present two case studies using our integrated PliMon -OSM implementation to demonstrate the salient aspects of our proposed approach.

1) Case Study 1: Metric Monitoring Frequency Adaptation: Here we assess how DE calculates PI and monitoring frequency in a more realistic environment. For this, we simulate CPU load on one of the connected VNFs. Throughout the experiment, we use 4 load profiles (no load, low, medium and high in that order), a metric of maximum CPU utilization over all cores and set the upper threshold at 60%.

Figure 6 shows what monitoring samples are taken (red dots) for an underlying evolution of CPU utilization (blue line). The zone in red indicates an undesirable state of the system, *Danger Zone*, during which monitoring samples have to be collected frequently as we want more information why the system ended up in that state. The green area indicates the PI for a chosen monitoring period at a confidence level of 95%.

In Figure 6 we also see that at first the system does not adjust the monitoring frequency, which stays at 1 sample every 10 seconds until sufficient number of samples are collected to reliably calculate the metric variability. After this initial period, we gradually decrease frequency to 1 sample every 50 seconds (note that this is the least frequency: we do not decrease it further so as to spot deviations in a timely manner). As can be seen in a period where PI does not cross the red zone, we space out monitoring points and when it does we monitor frequently until the PI drops back down again.

2) Case Study 2: Multi-Tier Monitoring: This experiment demonstrates that the metric tiers function as intended. Here we use two tiers of metrics: Tier 0 and Tier 1. In Tier 0, we place *ping* metric from eNodeB and, in Tier 1, we place the basic compute metrics from all the network functions (CPU, memory and network utilizations). The *ping* metric here indicates a delay

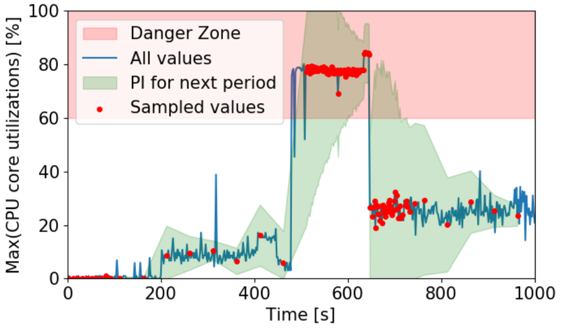

Figure 6. Example illustrating monitoring frequency adaptation of max CPU core utilization metric.

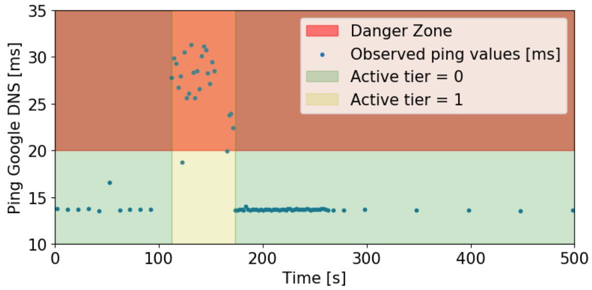

Figure 7. Multi-tier monitoring in action. Also shown is monitoring frequency adaptation of Tier 0 metric (eNodeB to Google DNS ping) over time across normal and congested network conditions.

(in milliseconds) when pinging Google DNS (an inexpensive proxy for end-to-end Quality of Service (QoS)). Under the normal operating conditions the delay is very stable at around 13ms. As such, we set the upper threshold to 20ms providing headroom in normal conditions.

In Figure 7 it can be seen that only Tier 0 metric is collected at first when *ping* indicates that the QoS is acceptable. At 120s we simulate high-traffic congestion conditions by using speedtest-cli on eNodeB. This raises the *ping* values, which invokes monitoring of Tier 1 metrics. When Tier 0 is no longer in violation, we deactivate Tier 1 metrics but continue monitoring ping at high frequency as instructed by the DE. When the implied PI of the ping narrows down, the DE instructs to gradually decrease the *ping* monitoring frequency.

The benefit of PliMon is evident here as we get more data on sub-par network performance from Tier 1 (e.g.: network usage at eNodeB) while saving monitoring overheads by only having Tier 1 metrics monitored during Tier 0 violation.

V. Conclusions

In this work, we have presented PliMon , a monitoring system targeting virtualized mobile networks. PliMon provides the flexibility to obtain custom metrics from each VNF, including generic compute metrics (e.g., CPU/memory utilization), VNF specific metrics (e.g., number of active user connections, eNodeB related measures), or end-to-end service quality metrics (e.g., device-Internet latency). Furthermore, PliMon reduces the overall monitoring overhead by employing statistical time series inference to detect periods of stability for different features to reduce their monitoring frequency when possible, and by dividing various metrics into tiers based on their significance and dynamically adjusting the tiers monitored based on the observed violations. Through its integration with the OSM MANO platform and our experimental evaluation, we demonstrate the compatibility of PliMon with production ready MANO systems and potential significant savings in terms of the monitoring overhead.

As possible avenues for future work, we will consider more robust time series inference methods (e.g., ARIMA) and the ability to use multi-dimensional, non-linear threshold boundaries. We will also consider exploring the monitoring delays and the mean time from the occurrence of an anomaly until its detection. This would in turn improve the confidence in the assertions regarding PliMon's production performance.

References

[1] A. Sutton, "5G Network Architecture," *J. Inst. Telecommun. Professionals*, vol. 12, no. 1, pp. 9–15, 2018.

[2] ETSI, "Network Functions Virtualisation – White Paper on NFV priorities for 5G," https://portal.etsi.org/nfv/nfv_white_paper_5g.pdf, accessed: 10 Feb 2019.

[3] K. Samdanis *et al.*, "5G Network Slicing - Part 1: Concepts, Principles, and Architectures," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 70–71, Aug 2017.

[4] V. Sciancalepore, F. Yousaf, and X. Costa-Pérez, "z-TORCH: An Automated NFV Orchestration and Monitoring Solution," *IEEE Transactions on Network and Service Management*, vol. PP, pp. 1–1, 08 2018.

[5] OSM, "Official OSM website," https://osm.etsi.org/.

[6] ONAP, "ONAP Architecture," https://www.onap.org/wp-content/uploads/sites/20/2018/11/ONAP_CaseSolution_Architecture_112918FNL.pdf.

[7] OSM, "OSM Release 4 Whitepaper," https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseFOUR-FINAL.pdf, accessed: 10 Feb 2019.

[8] Microsoft, "Azure Monitor," https://docs.microsoft.com/en-us/azure/azure-monitor/overview, accessed: 2018-10-15.

[9] Zabbix, https://www.zabbix.com/, accessed: 2018-10-15.

[10] Nagios, https://www.nagios.org/, accessed: 2018-10-15.

[11] Nokia, "Nokia Wireless Network Guardian," https://networks.nokia.com/products/wireless-network-guardian.

[12] Amdocs, "Amdocs Deep Network Analytics," https://tinyurl.com/ya6ausmh, accessed: 2018-10-15.

[13] Y. Li *et al.*, "FlowRadar: A Better NetFlow for Data Centers." in *Nsdi*, 2016, pp. 311–324.

[14] M. Moshref *et al.*, "Trumpet: Timely and precise triggers in data centers," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 129–143.

[15] H. J. Syed, A. Gani, F. H. Nasaruddin, A. Naveed, A. I. A. Ahmed, and M. Khurram Khan, "Cloudprocmon: A non-intrusive cloud monitoring framework," *IEEE Access*, vol. 6, pp. 44 591–44 606, 2018.

[16] R. Hyndman, A. Koehler, K. Ord, and R. Snyder, "Prediction intervals for exponential smoothing using two new classes of state space models," *Journal of Forecasting*, vol. 24, no. 1, pp. 17–37, 2005.

[17] T. Finch, "Incremental calculation of weighted mean and variance," 2009.

[18] G. M. Yilma, F. Z. Yousaf, V. Sciancalepore, and X. Costa-Perez, "On the challenges and KPIs for benchmarking open-source NFV MANO systems: OSM vs ONAP," https://arxiv.org/abs/1904.10697, 2019.

[19] Openair-CN, https://gitlab.eurecom.fr/oai/openair-cn, 2018, accessed: 2018-09-11.

[20] N. Nikaein *et al.*, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM CCR*, vol. 44, no. 5, pp. 33–38, 2014.

[21] Michael Patrick Allen, *Understanding Regression Analysis*. Springer, Boston, MA, 1997.