An Intelligent, Time-Optimized Monitoring Scheme for Edge Nodes

Christos Anagnostopoulos

School of Computing Science, University of Glasgow, G12 8QQ, Glasgow, UK

Kostas Kolomvatsos

Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, 15784, Greece

---

Abstract

Monitoring activities over edge resources and services are essential in today's applications. Edge nodes can monitor their status and end users/applications requirements to identify their 'matching' and deliver alerts when violations are present. Violations are related to any disturbance of the desired *Quality of Service* (QoS). QoS values depend on a number of performance metrics and can differ among applications. In this paper, we propose the use of an intelligent mechanism to be incorporated in monitoring tools adopted by edge nodes. The proposed mechanism continually observes the realizations of performance parameters that result in specific QoS values and decides when it is the *right* time to 'fire' mitigation actions. Hence, edge nodes are capable of changing their configuration to secure the desired QOS levels as dictated by end users/applications requirements. In our work, a mitigation action could involve either upgrades in the current services/resources or offloading tasks by transferring computational load and data to peer nodes or the Cloud. We present our model and provide formulations for the solution of the problem. A high number of simulations reveal the performance of the proposed mechanism. Our experiments show that our scheme outperforms any deterministic model defined for the discussed setting as well as other efforts found in the respective literature.

Keywords: Edge Computing, Edge Nodes, Monitoring, Quality of Service,

1. Introduction

In a Cloud computing setting, end users try to acquire remote access to a number of resources. Resources include hardware as well as software applications. Cloud services provide specific functionalities to users over the underlying hardware infrastructures. However, the advent of various technologies like the *Internet of Things* (IoT) [7], *Mobile Edge Computing* (MEC) [3], *Network Function Virtualization* (NFV) [42] and *Software Defined Networking* (SDN) [31] challenges the Cloud Computing model. For instance, if we realize the data processing activities close to end devices, we can reduce the latency in the provision of the requested services (the use of Cloud resources is characterized by increased latency [12]). Hence, the paradigm of Fog and Edge computing come into the scene [61]. Both, Edge and Fog computing involve the transfer of storage and processing at the edge of the network, where data are collected. End users/applications asking for analytics can be served in the minimum time, thus, supporting their requirements for getting a response in real time. However, these two technologies are not identical. The OpenFog Consortium identifies that Edge computing is often erroneously called Fog computing [60]. The focus of this distinction is mainly based on the hierarchical relation between them and the services that can provide. Fog computing can provide computing, networking, storage, control, and acceleration at multiple points of the hierarchy starting from Cloud and reaching the end devices while Edge computing tends to be limited to the provision of services at the edge of the network [60]. In short, the differences that the OpenFog Consortium detects between the two technologies are [43]: (i) Fog works with the Cloud while Edge is defined by the exclusion of Cloud; (ii) Fog is hierarchical, where Edge tends to be limited to a small number of layers; (iii) In additional to computation, Fog also addresses networking, storage, control and acceleration. The problem in Edge computing is the limited computational capabilities of *Edge Nodes* (ENs) compared to

the Cloud infrastructure and Fog nodes. Usually, the storage and processing can be performed in small edge servers or even more in end devices themselves. The critical issue is that ENs are connected with a high number of devices, thus, making imperative the need of monitoring their performance in order to efficiently support end users/applications processing tasks and data collection.

A monitoring service is the key element of any management system [1]. The reason is that management systems try to automate various processes in the infrastructure, thus, an autonomous module should supervise the independent components. In any case, services executed in ENs should be characterized by high *Quality of Service* (QoS) to efficiently support end users/applications. QoS characteristics may involve the performance of the underlying hardware as well as the performance of the execution of specific software solutions. In the relevant literature, one can find definitions for a number of parameters affecting QoS like CPU performance [22], response time, completion time, throughput, network utilization, mean-time between failures, mean-time to switchover, mean-time system recovery, etc [28], [37], [41], [57]. However, meeting the desired QoS levels does not depend only on the provided services. It also depends on users'/applications' requirements that could be updated over time. When a user/application adopts a 'public' EN to host or consume a critical service, performance variability and availability become a major concern [2]. Therefore, monitoring tools at ENs are essential in maintaining QoS at high levels and sustaining the performance of every application [6].

So far, monitoring services are offered by Cloud providers or third party companies. In Edge computing, the functionalities that should be adopted by such monitoring mechanisms are related to specific performance metrics for the software/hardware and users/applications requirements. Alerts can be produced to inform ENs for possible malfunctions and the need for applying a mitigation plan. In this paper, we propose a model to be adopted in a monitoring module that will be responsible to deliver alerts when potential QoS violations could be present in an EN. The proposed scheme is 'node-' and 'user/application-oriented' at the same time. If an EN cannot efficiently support the desired

level of QoS, a mitigation action should be initiated. Such an action may lead to the offloading of processing tasks and data to Cloud, Fog or peer nodes, if possible [29]. The 'user/application-oriented' aspect of our scheme deals with the monitoring activities performed to detect updates in users'/applications' requirements relying on ENs to enjoy the provided services. The discussed mechanism works in a *pro-active* manner trying to handle the problem before it is realized in contrast to the majority of the available monitoring mechanisms.

We assume that the mechanism is connected with a component that results in a QoS realization based on the set of values for the performance metrics under consideration. The mechanism should wait to secure that QoS is not violated. Nonetheless, the decision should be immediate as users/applications desire to have a high QoS and an uninterrupted execution. In this paper, we adopt the principles of *Optimal Stopping Theory* (OST) [44] to determine the *right* time for a meaningful decision on a mitigation action over sequentially observed QoS values. We model the discussed scenario and build an *optimal stopping decision making mechanism*. Based on the sequentially received QoS values realized through a set of *Key Performance Indicators* (KPIs), our mechanism identifies the appropriate time to stop the monitoring process and 'fire' a mitigation action. As noted, our aim is to update the configuration of ENs, offload the specific task or any other action that will secure the QoS at high levels. The significant is that our mechanism tries to 'match' the performance of an EN with the updates in the requirements of users/applications, thus, to pro-actively take the appropriate measures. We propose an efficient mechanism that maintains the status of ENs to the appropriate one to continuously meet end users/applications requirements. In the following list, we present the contributions of our work:

- we support a monitoring mechanism to be applied at the edge of the network and assist ENs to monitor their performance and end users/applications requirements;
- we provide a decision making mechanism for ENs that results the appro-

priate time for applying mitigation actions towards securing the QoS at high levels;

- we adopt the principles of OST and present a generalized optimal stopping rule for firing mitigation actions. Our model focuses on time-optimized decisions, i.e., to find the appropriate time to apply mitigation actions. ENs should delay their decision to collect more contextual data, however, should not wait for long as any heavy disturbance of QoS will limit the performance of applications;

- we propose a mechanism responsible to continuously 'align' the performance of ENs with end users/applications requirements. The proposed model takes into consideration both aspects of the problem, thus, it tries to 'match' them and take the most appropriate decision;

- we provide an experimental evaluation of our model and a comparative assessment targeting to reveals its pros and cons.

The proposed mechanism could be combined with a system responsible to handle the life cycle of applications. This combination is beyond the scope of this paper and left for future work.

The paper is organized as follows. The related work is presented in Section 2 while in Section 3, we present our scenario and describe our setting. In Section 4, we analytically describe the proposed mechanism. In Section 5, we present the performance of our mechanism based on a high number of simulations. Finally, in Section 6, we conclude our paper by describing future extensions of our work.

2. Related Work

In recent years, researchers' attention is paid on the delivery and the combination of new services over the available infrastructure. The main focus is on providers' side meaning that main research results are related to the performance of services. Many efforts focus on services composition [5], [8], [13], [16],

[18], [30], [33], [48], [59], [62], while others focus on resource selection and deployment in Cloud [9], [45], [47]. Semantic techniques, planning algorithms and evolutionary approaches are some of the adopted techniques. The aim is to have new services through a combination of already provided, some times incompatible, components. The performance of such 'complex' services depends on the performance of their parts. The most difficult scenario is when these parts are offered by different providers. This is more intense when a user wants to monitor the performance of services to secure that the QoS is at high levels. The reason is that users should monitor a number of metrics for a set of providers. One can find commercial and open source tools offering Cloud services monitoring. An extensive survey in the field is available in [2]. Apart from these tools, users can build their own models. However, significant attention should be paid on the underlying resource management. An extensive survey on the Cloud resource management techniques can be found in [38].

In our review of the relevant literature, we start with the virtual resources monitoring in Cloud. For a full survey, the interested reader cal refer in [52]. The approach of providing Monitoring-as-a-Service (MaaS) [4] is usable for both providers and end users. Such an approach offers the monitoring software as a part of the Cloud infrastructure capable of performing monitoring activities even in federated Clouds. The requirement is to have all the involved infrastructures applying the same monitoring mechanism. Other approaches involve dashboards where users can observe the collected performance metrics realizations [32]. Usually, such tools are devoted to monitor the available hardware like the CPU, memory, storage and so on and so forth. Software agents can be also adopted for monitoring purposes [40]. Agents can undertake the responsibility of observing the performance of specific metrics and transfer the relevant data to an application that will decide if a potential QoS violation is present. In any case, the centralized approach in the decision making adds obstacles towards having a mechanism that is fully adapted to the needs of any local processing unit. Lattice, proposed in [14], aims at providing observation functionalities over virtual infrastructures. The framework provides an API and the necessary

interfaces to be adopted by software components that may want to support customized solutions. iOverbook, presented in [11], is an intelligent, autonomous tool capable of supporting monitoring activities in virtualized, however, heterogeneous environments. The basis of the tool is the decision making on top of a neural network that is responsible to deliver forecasting of the resources' usage. The tool can have a view on the future trends of the usage, thus, it can act proactively in identifying any potential problems.

Another form of virtualization that demands less resources are containers [50]. Containers can be uploaded in different hosts without the requirement of adopting a specific operating system. In Edge computing, there is the need of monitoring such container based installations at the ENs. In [51], the authors propose a distributed self-adaptive model that is applied on the Edge computing setting to ensure the QoS in time critical applications. The same image can be uploaded in different locations. The evaluation of docker containers is the subject of [46]. The performance of dockers is evaluated in terms of the hardware performance approximating the performance of a native environment. Securing the QoS levels in time critical applications on top of dockers is the focus of [19]. A priority scheme is adopted to manage the access to the network for any docker. The main focus is paid on scenarios where multiple dockers are present in the same host.

Additionally, a set of efforts focuses on the monitoring of applications performance [20], [21], [27], [34], [39], [49], [53], [58]. Such frameworks mainly aim to observe specific KPIs in a distributed manner facilitating scale in/out actions to be aligned with end users requirements. Applications can be separated into a number of parts hosted in different e.g., containers. The management of this distributed nature of the approach should be accompanied by a powerful coordination mechanism. Meeting end users requirements is imperative in environments where time critical applications should be provided. A proactive mechanism based on intelligent techniques (e.g., neural networks) may be applied and satisfy users future demands [26]. Performance data can be seen as time series and their processing should be concluded in real time. Such data

cover the entire set of 'points' where the attention should be paid, e.g., the CPU performance, the remaining storage capacity, the memory usage and so on and so forth.

Monitoring systems are also proposed to be adopted by the edge infrastructure trying to deal with several challenges in the domain. Edge computing has to do with a very dynamic infrastructure where nodes may join and leave the network frequently. In addition, tasks scheduling and resource provisioning require monitoring KPIs with very low and predictable latency to make fast decisions [10]. To reduce the latency, instead of sending the monitored data to the Fog or Cloud, ENs may store them for local processing. Such an approach may assist in the efficient management of data and performance intensive applications like online games and Augmented Reality. In such cases, applications endpoints should be close to end users together with the management of location context [15]. In [56], the authors propose a monitoring approach for Fog systems in a cyber-manufacturing scenario incorporating multiple technologies like Wireless Sensor Networks (WSNs), communication protocols, and predictive analytics. PyMon [23] is another monitoring framework fully aligned with the edge infrastructure needs. The proposed system is lightweight designed to be hosted by nodes with limited computational capabilities. The authors of [10] present the requirements of edge monitoring tools and propose the FMonE framework. This framework allows the deployment of monitoring workflows in the edge infrastructure. In [24], the authors propose a scheme for the detection of key nodes and their influence on the status of the entire network. Based on these nodes, the proposed approach tries to identify links that are not monitored, thus, it is able to setup monitoring activities in key locations of the network. The Edge NOde Resource Management (ENORM) framework, presented in [54], aims at the provisioning of auto-scaling edge resources. The auto-scaling actions are a type of mitigation actions when increased resources should be devoted to efficiently service applications. The benefits are related to a reduced latency and reduced data transfer between ENs and Cloud. Finally, in [55], the authors discuss an optimization platform that deals with the end-to-end throughput in real

time. The optimal throughput is achieved by adopting a set of components, i.e., a dynamic routing engine, a performance monitoring scheme and an information exchange model.

To the best of our knowledge, the available monitoring tools (mainly proposed to be used in Cloud) do not offer an automated intelligent mechanism that assists ENs to decide when it is the appropriate time to proceed to a mitigation action (*pro-active action*). The majority of the related efforts deal with mechanisms that provide alerts when violations in specific constraints are present. However, the envisioned pro-active response is very important in the case of composite services as multiple providers are involved, thus, the monitoring tool should take decisions based on a large number of KPIs originated in different providers.

3. Rationale and Preliminaries

3.1. Performance Monitoring

Each user/application c_i selects a service or a set of services (e.g., data collection service, data processing tasks) offered by an EN p_k. We focus on the monitoring process of a specific service s_j offered by p_k. Our mechanism will monitor the status of a *service assignment* $\mathrm{A}(c_i, s_j)$ which connects c_i with s_j. Each $\mathrm{A}(c_i, s_j)$ is configured with the c_i profile and s_j end reference point. We have to notice that our model can be applied in any setting that monitors the aforementioned assignments. The mathematical analysis that we provide in this effort will be the same, however, the adopted assumptions and parameters are aligned with the Edge computing infrastructure. In Edge computing, assignments are more dynamic compared to the remaining cases (i.e., Fog and Cloud). Monitoring needs differ as edge based resources are highly dynamic and the executed tasks exhibit a short lifecycle while being frequently instantiated [1]. In addition, migration activities may need to be performed more frequently compared to Cloud and Fog. Finally, IoT devices mobility can also affect the services execution at the edge. All these issues affect the envisioned assignments making imperative the need for our monitoring mechanism. QoS violations may

be more frequent making us to be based on the specific assumptions and parameters as defined throughout our paper.

Our system registers $A(c_i, s_j)$ and defines an 'observer' responsible to receive QoS values for $A(c_i, s_j)$. The 'observer' has two interfaces: (a) with end users/applications and (b) with a component resulting the final QoS value based on a specified set of KPIs. The 'observer' monitors any changes to users/applications' requirements (e.g., increased data delivery rate, demand for additional services) or in services performance (e.g., latency for delivering data processing results). The aim is to have an insight of *when* the assignment $A(c_i, s_j)$ *will be* 'violated'. The assignment $A(c_i, s_j)$ is 'violated' when:

- the system observes user/application requirements and 'sees' that the user/application requires a higher or a lower QoS compared to the current setting. For instance, if the user/application needs a low QoS, the system could devote less resources and 'migrate' resources to other users/applications.

- the system observes the performance of services and decides when the desired QoS level is not fulfilled. For each application, we can easily define a QoS threshold based on the specific requirements. For instance, if we focus on Augmented Reality applications, QoS should reach the upper bound (unity, if we consider that QoS is in the interval [0,1]).

3.2. Problem Formulation

We assume that the time is divided in slots of duration d, e.g., the *sampling* interval could be $d = 1$ time unit. The 'observer', at each time slot, receives values for the desired KPIs and determines the final QoS. That is, at each time slot, a QoS value is calculated, thus, we obtain a sequence of QoS values. Without loss of generality, we consider the parameter QoS_v indicating the QoS violation for a specific service. QoS_v is represented through a non-negative integer, which is assumed to be Poisson distributed with parameter λ. The QoS_v value is desired to be as small number as possible, with $QoS_v = 0$ representing the highest QoS level. The QoS_v value, S_n, is observed in the interval $((n -$

$1)d, nd]$, $n > 0$ is s with probability

$$P_d\{S_n = s\} = \frac{e^{-\lambda d}(\lambda d)^s}{s!}.$$

We now consider the stochastic process X_n, $n \geq 0$, which represents the **accumulated** QoS_v values up to nd, i.e., $X_n = \sum_{k=0}^{n} S_k$. We set $X_0 = 0$ and for $n \geq 1$, we define

$$X_n = \begin{cases} x & \text{if the number of accumulated } QoS_v \text{ values } x \text{ in } [0, nd] \\ & \text{satisfy } x < Y = y \\ x_{\{y\}} & \text{if the number of accumulated } QoS_v \text{ values } x \text{ satisfy} \\ & x \geq Y = y \text{ at the end of the interval } [(n-1)d, nd] \\ & \text{whereas at the beginning of the aforementioned} \\ & \text{time interval } x < Y = y \end{cases}$$

The random variable Y represents an upper bound of the *acceptable tolerance* on the QoS_v. Specifically, Y indicates the acceptable sum of QoS_v values in which the QoS is considered satisfiable by the user/application. Once the sum of QoS_v exceeds this value, the mechanism should proceed with a mitigation decision. In general, Y is random, but can also be a fixed, pre-determined, value for each application.

$\{X_n\}$ is a Markov chain with states x_y; states x are transient. In the state space of the monitoring process, we define a function $g(\cdot)$ over the sum of QoS_v values that reflects the *cost* of the system at the end of the the n^{th} time slot. Specifically, in this paper we set (if $X_n = x$)

$$g(x) = ax - n\beta \tag{1}$$

whereas, if $X_n = x_y$,

$$g(x_y) = G(x, y) \tag{2}$$

Eq(1) accounts for the cost due to the cumulative QoS_v values with a factor $a > 0$, whereas, Eq(2) accounts for a gain due to postponing the mitigation process, with a factor of $\beta > 0$. The rationale behind the cost function is that the

mechanism after observing a sequence of (random) QoS_v values, $S_0, S_1, \ldots, S_n$, decides whether to stop the monitoring process at time slot n and proceed with a decision of mitigation due to violations of the corresponding assignment $\mathrm{A}(c_i, s_j)$ or, continue the service. However, since QoS values are random and the EN has defined an acceptable bound on the tolerance on QoS_v, the mechanism should deal with this stochasticity to proceed with an 'optimal' decision.

The parameter a indicates the 'penalty' that we incur if successive high QoS_v values (referring to low degree of quality - QoS) are accumulated with time. On the other hand, the β parameter indicates the gain of using the edge services as long as $\mathrm{A}(c_i, s_j)$ is not violated, or the received QoS_v values are within acceptable levels. This implies that the EN desires to postpone a possible mitigation decision. The meaning of β is that it represents the cost for a mitigation process, thus, once the mechanism decides at n not to proceed with a mitigation decision, a 'reward' of $-\beta$ incurs due to this postponing. Nonetheless, once a mitigation decision is needed, then we incur a cost, which might be a function of the current cumulative QoS_v values X_n (i.e., the up to now *behavior* of the invoked service in terms of QoS) and the tolerance on QoS_v represented by Y. This cost is expressed in Eq(2) and refers to the *maximum tolerance* that can be accepted by the EN. Hence, the problem is to find the right time (i.e., n^{th} time slot) to proceed with one of the two following decisions:

D1 Continue the invocation of the service and observe its QoS_v value at the next time slot $n+1$ without proceeding with a mitigation process with cost β;

D2 Stop the monitoring process and proceed with a mitigation action incurring a cost equal to β.

We formalize our problem as follows:

Problem 1. Given a tolerance Y on QoS_v values and a maximum tolerance $G(X_n, Y)$, with X_n be the cumulative sum of QoS_v values up to n, determine a right time (optimal stopping time), which minimizes the cost function in Eq(1) and Eq(2).

We treat Problem 1 through the OST, since the observed QoS_v values are random and a decision is desired for minimizing the rational cost defined in Eq(1) and Eq(2). The only information we have in our hands is QoS_v values observed up to n and the tolerance on QoS_v for the considered service. The derived optimal stopping time is that which minimizes the cost function $g(\cdot)$ based on the two above-mentioned decisions **D1** and **D2**.

3.3. Optimal Stopping Theory

The Optimal Stopping Theory (OST) [44], concerns finding the right time to take an action (decision) based on sequentially observed random variables. The final aim is to minimize the expected cost. The optimal stopping problem is defined by a sequence of random variables $X_1, X_2, \ldots$ whose joint distribution is known and a sequence of real-valued cost functions $g_0, g(x_1), g(x_1, x_2), g(x_1, x_2, x_3), \ldots$. Let (Ω, B, P) be the probability space, and $\mathcal{G}_t$ be the sub-σ-field of B generated by $X_1, \ldots, X_n$. We have a sequence of σ-fields as $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \ldots \mathcal{G}_n \subset B$. A *stopping time* is defined as a random variable $N \in \{0, 1, \ldots, \infty\}$ such that the event $\{N = n\}$ is in $\mathcal{G}_n$. The aim is to choose an *optimal stopping time* n^* to minimize the expected cost $E[g_{n^*}]$. If there is no bound on the number of steps at which one has to stop, this is an infinite horizon problem and the optimal return can be computed via an *optimal stopping rule*.

4. Time-Optimized Performance Monitoring

4.1. A generalized optimal stopping rule

In the following, we provide the optimal stopping rule for Problem 1. Specifically, we find the optimal policy of postponing the mitigation action due to 'good' QoS_v values for Problem 1.

We first describe the transition probabilities for the Markov chain $\{X_n\}$. Let us denote $\mathcal{P} = P_d\{s\}\frac{\bar{F}_Y(x+s)}{\bar{F}_Y(x)}$. Then, we obtain

$$\begin{aligned} &p\{x+s \,|\, x\} = \mathcal{P}, && 0 \le s, \quad x+s < y \\ &p\{(x+s)_{\{y\}} \,|\, x\} = P_d\{s\}\tfrac{P\{Y=y\}}{\bar{F}_Y(x)}, && x < y \le x+s \\ &p\{x_{\{y\}} \,|\, x_{\{y\}}\} = 1, && y \le x \end{aligned} \tag{3}$$

where $Y = y$ is a realization value of Y and $\bar{F}_Y(x) = 1 - F_Y(x)$ (cumulative distribution function of Y), that is

$$\bar{F}_Y(x) = \sum_{m=x+1}^{\infty} P\{Y = m\}$$

$$\bar{F}_Y(x+s) = \sum_{\ell=x+s+1}^{\infty} P\{Y = \ell\}$$

The previous transition probabilities should sum to unity given that $X_n = x$. For $X_n = x_{\{y\}}$ it is obvious. For $X_n = x$, we have that

$$\begin{aligned}
&\sum_{s=0}^{\infty} p\{x+s \,|\, x\} + \sum_{s=0}^{\infty} \sum_{y=x+1}^{x+s} p\{(x+s)_{\{y\}} \,|\, x\} \\
&= \sum_{s=0}^{\infty} P_d\{s\} \frac{\bar{F}_Y(x+s)}{\bar{F}_Y(x)} + \sum_{s=0}^{\infty} \sum_{y=x+1}^{x+s} P_d\{s\} \frac{P\{Y=y\}}{\bar{F}_Y(x)} \\
&= \sum_{s=0}^{\infty} \mathcal{P} + \sum_{s=0}^{\infty} P_d\{s\} \sum_{y=x+1}^{x+s} \frac{P\{Y=y\}}{\bar{F}_Y(x)} \\
&= \sum_{s=0}^{\infty} \mathcal{P} + \sum_{s=0}^{\infty} P_d\{s\} \frac{\bar{F}_Y(x) - \bar{F}_Y(x+s)}{\bar{F}_Y(x)} \\
&= 1
\end{aligned}$$

The expected cost at $X_n = x$ is given by

$$C(x) = E_Y[g(x)] = \sum_{m=x+1}^{\infty} g(x) \frac{PY = m}{\bar{F}_Y(x)} \tag{4}$$

whereas, at $X_n = x_y$, the expected cost is simply

$$C(x_y) = g(x_y) = G(x, y). \tag{5}$$

Let now $\mathcal{C}(x)$ represent the expected cost under an optimal policy given that we take a decision at x. Using the *one-stage look ahead rule* [44], $\mathcal{C}(x)$ should satisfy the following functional equation (optimality equation):

$$\mathcal{C}(x) = \min\{C(x), P\mathcal{C}(x)\}. \tag{6}$$

The operator P denotes that one more step is taken by the mechanism, that is, we postpone the mitigation action in the next interval. Thus,

$$PC(x) = \sum_{s=0}^{\infty} C(x+s) p\{x+s \,|\, x\} + \sum_{s=0}^{\infty} \sum_{y=x+1}^{x+s} C((x+s)_{\{y\}}) p\{(x+s)_{\{y\}} \,|\, x\} \tag{7}$$

The rationale behind Eq(7) is that it is optimal to stop (decision **D2**) at the current time where we have accumulated x, if $C(x)$ is less or equal to $PC(x)$. Otherwise, it is optimal to continue the monitoring process, i.e., decision **D1**.

Theorem 1. If at n, QoS_v values sum up to X_n, the optimal stopping rule for Problem 1 is provided by:

$$\sum_{m=X_n+1}^{\infty}\sum_{s=m-X_n}^{\infty}(G(X_n+s,y)-a(X_n+s)+(n+1)\beta)P_d\{s\}\frac{P\{Y=m\}}{\bar{F}_Y(X_n)} \geq \beta - a\lambda d.$$

Proof. Substituting (5), (6) in (8) we obtain

$$PC(x)=\sum_{s=0}^{\infty}\sum_{m=x+s+1}^{\infty}(a(x+s)-(n+1)\beta)\frac{P\{Y=m\}}{\bar{F}_Y(x+s)}\mathcal{P}+\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}G(x+s,y)\mathcal{R},$$

where $\mathcal{R}=P_d\{s\}\frac{P\{Y=m\}}{\bar{F}_Y(x)}$. Therefore,

$$\begin{aligned}
PC(x) &= \sum_{s=0}^{\infty}\sum_{m=x+s+1}^{\infty}(a(x+s)-(n+1)\beta)\frac{P\{Y=m\}}{\bar{F}_Y(x)}P_d\{s\}+\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}G(x+s,y)\mathcal{R}\\
&= \sum_{s=0}^{\infty}\sum_{m=x+1}^{\infty}(a(x+s)-(n+1)\beta)\mathcal{R}-\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}(a(x+s)-(n+1)\beta)\mathcal{R}\\
&\quad+\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}G(x+s,y)\mathcal{R}\\
&= \sum_{s=0}^{\infty}\sum_{m=x+1}^{\infty}(ax-n\beta)\mathcal{R}+\sum_{s=0}^{\infty}\sum_{m=x+1}^{\infty}(as-\beta)\mathcal{R}\\
&\quad+\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}(G(x+s,y)-a(x+s)+(n+1)\beta)\mathcal{R}\\
&= C(x)+a\lambda d-\beta+\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}(G(x+s,y)-a(x+s)+(n+1)\beta)\mathcal{R}.
\end{aligned}$$

Therefore, it is optimal to stop if the following condition holds true:

$$\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}(G(x+s,y)-a(x+s)+(n+1)\beta)\mathcal{R}\geq\beta-a\lambda d \tag{8}$$

Changing the sum limits $\sum_{s=0}^{\infty}\sum_{m=x+1}^{x+s}(\cdot)=\sum_{m=x+1}^{\infty}\sum_{s=m-x}^{\infty}(\cdot)$ results in the following criterion:

$$\sum_{m=x+1}^{\infty}\sum_{s=m-x}^{\infty}(G(x+s,y)-a(x+s)+(n+1)\beta)\mathcal{R}\geq\beta-a\lambda d. \tag{9}$$

□

The criterion in inequality (10) refers to the optimal stopping rule, at which the mechanism makes decision **D2** in a time slot n where we have accumulated QoS_v values $X_n = x$ with which the inequality (10) holds true. Otherwise, it is optimal to continue, i.e., decision **D1**.

4.2. A simplified optimal stopping rule

In the discussed setting, we should specify $G(x, y)$ and $P\{Y = m\}$. For further simplifications, we assume a fixed maximum tolerance and a fixed acceptable tolerance upper bound on QoS_v. That is, we set $G(x, y) = G$ and $P\{Y = y\} = 1$ if $y = D$; otherwise 0, with $G > 0$ and $D > 0$. In this case, the optimal stopping rule of inequality (10) is simplified to:

$$\sum_{s=D-x}^{\infty} (G - a(x + s) + (n + 1)\beta)P_d\{s\} \geq \beta - a\lambda d \tag{10}$$

with $x + 1 < D < \infty$.

Since $X_n = X_{n-1} + S_n$, we can define the recursion:

$$\sum_{s=D-X_n}^{\infty} P_d\{s\} = \sum_{s=D-X_n}^{D-(X_{n-1}+1)} P_d\{s\} + \sum_{s=D-X_{n-1}}^{\infty} P_d\{s\}$$

with $X_0 = 0$, thus, initially, we obtain

$$\sum_{s=D}^{\infty} P_d\{s\} = 1 - \sum_{s=0}^{D-1} P_d\{s\}.$$

Moreover, since $\sum_{s=0}^{\infty} sP_d\{s\} = \lambda$, in a similar way, we can define the recursion

$$\sum_{s=D-X_n}^{\infty} sP_d\{s\} = \sum_{s=D-X_n}^{D-(X_{n-1}+1)} sP_d\{s\} + \sum_{s=D-X_{n-1}}^{\infty} sP_d\{s\}$$

with $X_0 = 0$, thus, initially, we obtain

$$\sum_{s=D}^{\infty} sP_d\{s\} = \lambda - \sum_{s=0}^{D-1} sP_d\{s\}.$$

Based on the above recursions, the criterion in inequality (10) for stopping at X_n is written as:

$$[G - aX_n + (n+1)\beta] \sum_{s=D-X_n}^{\infty} P_d\{s\} - a \sum_{s=D-X_n}^{\infty} sP_d\{s\} \geq \beta - a\lambda d. \quad (11)$$

Based on the above analysis, the simplified algorithm that our 'observer' adopts to instruct the initiation of mitigation actions is as follows:

Algorithm 1: The monitoring algorithm

input : $A(c_i, s_j)$

output: The invocation of mitigation actions

$n = 0$;

$X_n = 0$;

while *true* **do**

  $n++$;

  $QoS = \text{getQoS}(A(c_i, s_j))$;

  $S_k = \text{calculateQoSViolation}(QoS)$;

  $X_n \mathrel{+}= S_k$;

  if *Inequality (10) is satisfied* **then**

    Decision **D2** ;

    $n = 0$;

    $X_n = 0$;

  else

    Decision **D1** ;

  end

end

5. Performance Assessment

We report on the performance of the proposed mechanism through a set of simulations. We provide numerical results and give an insight on the strengths and weaknesses of the proposed model.

5.1. Performance Metrics & Simulation Set-up

Initially, we compare the proposed time optimized mechanism with a *Deterministic Stopping Rule* (DSR) in order to demonstrate the optimality achieved by our *Optimal Stopping Rule* (OSR) in Section 4.2. Specifically, a mechanism that is based on a DSR proceeds with a stopping decision (i.e., decision **D2**) iff the accumulated QoS_v values up to time slot n, $X_n = x$, exceeds a fixed threshold z. The DSR then:

D1 Continues the monitoring process at the next time slot $n+1$, if $X_n < z$;

D2 Stops and proceeds with a mitigation process, if $X_n \geq z$.

We evaluate our OSR with the DSR for diverse values of z, $z \in [1, D)$, to examine the cases where OSR results in better optimization of the $g(\cdot)$ function than the DSR. The evaluation of each mechanism (OSR and DSR) refers to the expected cost $E[g_n]$ with $g_{n^*} = aX_{n^*} - n^*\beta$ for the OSR, where n^* is the optimal stopping time at which inequality (10) holds true, and $g_{n^\#} = aX_{n^\#} - n^\#\beta$ for the DSR, where $n^\#$ is the stopping time based on the above-mentioned deterministic rules, respectively.

In addition, we compare our model with schemes found in the relevant literature and more specifically, with three other schemes adopted for predicting future QoS values. In their recent study, the authors of [25] present an analysis of multiple technologies aiming at the prediction of services QoS violation. Among them, they propose the use of: (i) the *Exponential Smoothing Model (ESM)*; (ii) the *Moving Average Model (MAM)*; and, (iii) the *Holt-Winter Double Exponential Smoothing Model (DESM)*. For comparing our *OST Model* (OSTM) with those three schemes, we rely on widely used classification KPIs, i.e., precision ϵ, recall ζ and accuracy ϕ. With the adoption of these metrics, we aim at revealing if the aforementioned models are capable of detecting QoS violations as generated by our synthetic trace. The discussed metrics are defined as follows: $\epsilon = \frac{TP}{TP+FP}, \zeta = \frac{TP}{TP+FN}, \phi = \frac{TP+TN}{TP+TN+FP+FN}$ where T refers to 'true', P refers to 'positive', F refers to 'false', and N refers to 'negative'. For instance, TP

refers to true positive events, i.e., identified events that had to be identified, FP refers to false positive events, i.e., identified events that had to not been identified, and so on and so forth.

The QoS_v value at each time slot is assumed to be Poisson distributed with parameter λ. However, the proposed optimal stopping mechanism can deal with any arbitrary QoS_v distribution; one can simply replace the $P_d\{s\}$ in Theorem 1 with the desired probability distribution. We deal with the Poisson distribution since through the λ parameter, we can *generate* QoS_v referring to low values, i.e., high degree of quality (QoS), by adopting a low λ value (e.g., $\lambda = 1$), or QoS_v values corresponding to a low degree of quality (QoS) by adopting a high relatively λ value. We experiment with $\lambda \in \{1, 5, 10\}$, representing low, medium, and high degree of quality (QoS). Through this approach, we can examine the behaviour of the proposed mechanism in dealing with a range of QoS_v patterns. Moreover, we experiment with difference mitigation cost values, i.e., β values, so as to demonstrate the eagerness of the mechanism to decide on either postponing or proceeding with a mitigation process. Obviously, as it will be shown, a high β value along with a stream of low QoS_v values (i.e., high degree of quality) results in postponing a mitigation decision, since the mechanism observes that the QoS_v, and QoS respectively, of the invoked service, is within the acceptable levels and an immature decision on a mitigation process would cost a lot the EN. On the other hand, once consecutive QoS_v values are relatively high, i.e., low degree of quality, the mechanism should decide on a mitigation process provided that β is relatively small. Furthermore, we experiment with different G values to examine the capability of the proposed mechanism in dealing with a high risk in terms of receiving a low QoS, referring to consequent unacceptable QoS values (high QoS_v values). This denotes whether the mechanism is able to proceed with optimal decisions either **D1** or **D2** to secure QoS_v and, respectively, QoS values within acceptable levels. We set $D = 100$ in experiments with $G = f \cdot D$, with $f \in \{10, 30, 60\}$, $a = 1$, and $\beta \in \{25, 50, 100, 200\}$. Without lot of generality, we consider the sampling period $d = 1$ time unit. For each experiment, we execute 1,000 runs and take the **Optimal Expected Cost (OEC)** and the

Deterministic Expected Cost (DEC) for the OSR and DSR mechanisms, respectively. The z deterministic threshold for the DSR takes values $z \in [1, D)$.

5.2. Performance Assessment

In Fig. 1, we plot OEC and DEC values for different λ; note that OEC is independent of z. The smaller the values are, the higher the reward is for remaining at the same setup. In these cases, the system receives low a QoS_v and respectively, it gains high quality services (QoS). The DSR mechanism performs worse than the OSR, especially for $z \leq 90$. This stands for $\lambda = 1$. When the mechanism receives high quality services, the appropriate decision is to remain at the same setup and avoid the mitigation process. The greater the λ is, the smaller the reward becomes for remaining at the same setup. This is natural as the mechanism receives consecutive high QoS_v values and respectively gains low quality services. Additionally, when $\lambda = 10$, the DSR and the OSR have similar performance when $z \geq 65$. In the discussed scenario, the difference of the performance between the OSR and the DSR mechanisms (we consider the minimum DEC value which corresponds to the maximum performance) is 6.93%, 7.27% and 1.33% for $\lambda = 1$, $\lambda = 5$ and $\lambda = 10$, respectively. Note that, there is a region of z values close to D in which the DSR approaches the OSR in terms of cost. Specifically, as $z \to D$ the DSR relaxes its tolerance, thus, it delays a mitigation decision. This results in minimizing the cost since n assumes relatively high values. However, given the stochasticity of QoS values, as long as $z \to D$, there is a high likelihood that a 'next' QoS_v value will result in exceeding the D tolerance, thus, incurring a cost of G. This indicates the incapability of the DSR to handle the stochastic nature of QoS in decision making, thus, resulting in high cost values close to the tolerance D. On the other hand, OSR takes into consideration the random nature of QoS_v through the optimal stopping rule. OSR relies its decision making on the behaviour/trend of the sequence of QoS_v through the cumulative sum X_n. In this context, OSR achieves the minimization of the cost function by using random stopping times (and not fixed as defined in DSR) that are governed by the stochasticity of QoS_v, as presented

in Theorem 1.

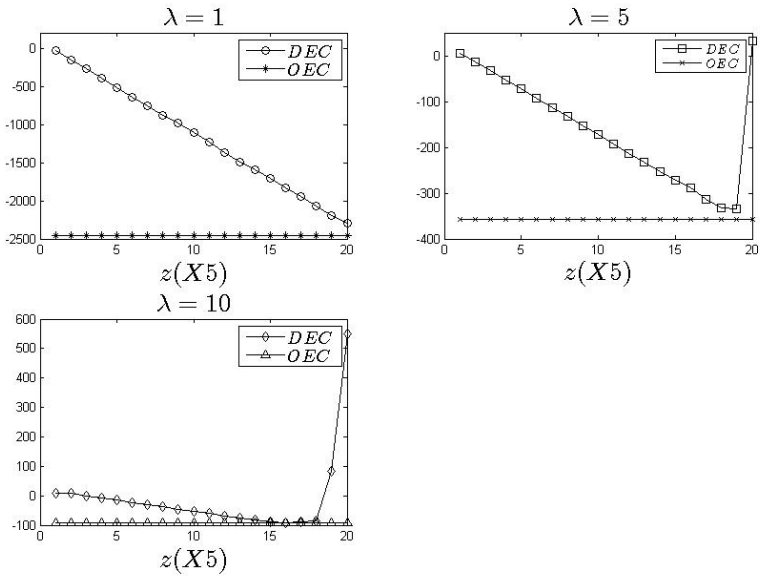

Figure 1: OSR vs. DSR for different λ.

In Fig. 2, we plot our results for different β values. Recall that β represents the mitigation cost. We observe similar results as in Fig. 1. The OEC defines the optimal limit as already discussed. The difference in the performance is significantly high especially when $z \leq 85$. It is worth noting that when $\beta = 50$ and $z \in [85, 90]$, the DSR mechanism performs better than the OSR. In this case, we obtain 4.95% reduction in the optimal expected cost. In the remaining cases ($\beta \in \{25, 100, 200\}$), we obtain 7.27%, 3.15% and 12.89% higher values (performance) from the OSR mechanism.

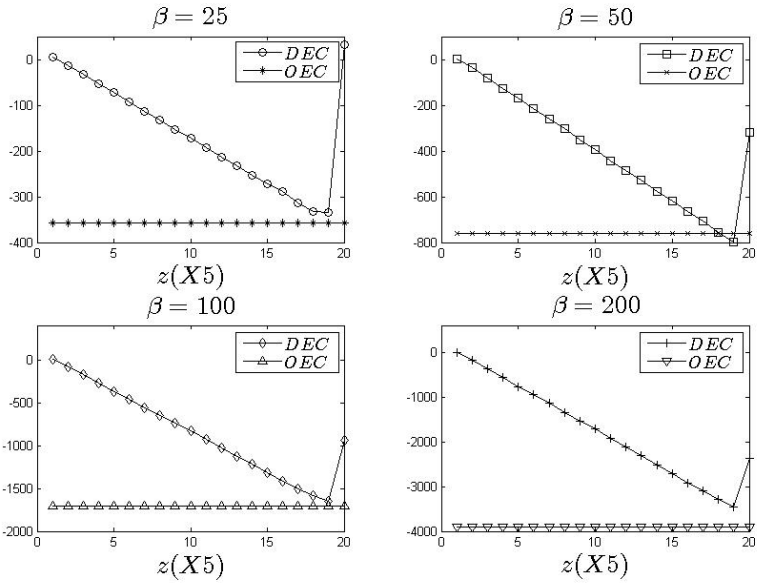

Figure 2: OSR vs. DSR for different β.

In Fig. 3, we experiment with various tolerance values. Similar results are also obtained for this set of experiments. The OSR mechanism performs better than the DSR mechanism for the majority of z values. When $G = 30$ and $z = 90$, the DSR achieves a better value compared to the OSR mechanism. The difference in the performance between the OSR and the DSR mechanisms is 3.15%, -2.17% and 0.04%.

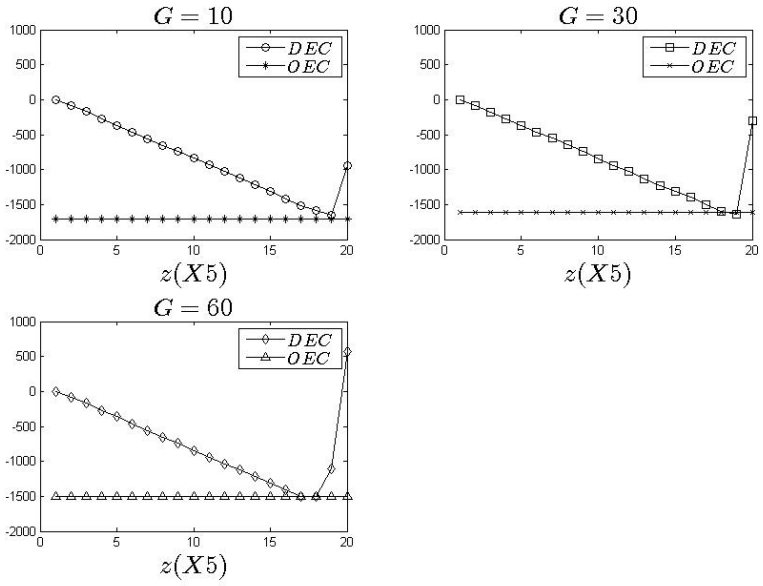

Figure 3: OSR vs. DSR for different G.

In general, the proposed OSR mechanism minimizes the expected cost of the mitigation process as revealed from our experiments in terms of high, medium and low QoS_v values, as well as when dealing with low and high mitigation costs and maximum tolerance values. On average, for the entire set of our experiments, the OSR achieves 3.49% smaller expected cost compared to the best achieved expected cost value obtained from any deterministic rule.

In Fig. 4, we provide plots for the probability density function (pdf) of n^* (the optimal stopping time for OSR) and for different λ. We observe that the smaller the λ is, the greater the n^* becomes. This is natural, as small λ means that the mechanism receives low QoS_v values and, respectively, it enjoys high quality services (high QoS). Hence, the EN should remain at the same setup and avoid any mitigation action. When $\lambda = 10$, users/applications enjoy low quality services (low QoS), thus, the mechanism results a mitigation action. In this case, n^* values are around 10 which is 89.5% lower than n^* values for $\lambda = 1$.

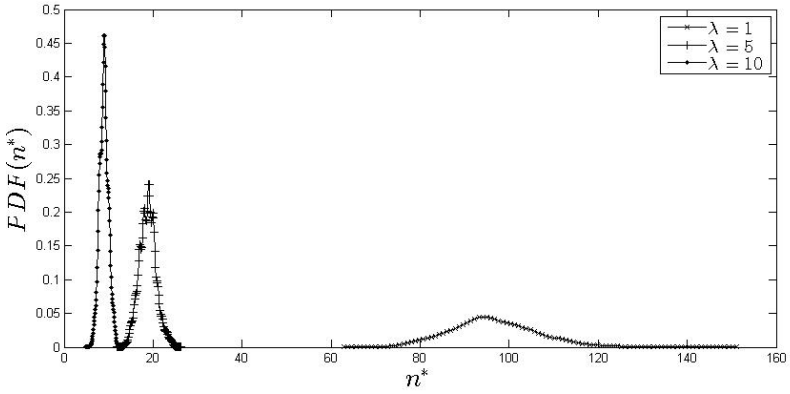

Figure 4: n^* pdf for the OSR mechanism.

In Fig. 5 and 6, we depict the pdf of $n^{\#}$ for the DSR model ($z = 10$ and $z = 50$ respectively). We observe a similar performance as in the OSR case. The $n^{\#}$ values are affected, as natural, by λ. A high λ leads to low $n^{\#}$, thus, a mitigation action is the immediate decision. The interesting is that a low z leads to low $n^{\#}$ as the threshold is immediately violated and the system decides the mitigation action. This observation stands for all λ realizations. Low z results a very low $n^{\#}$ (below 15) which means that the system decides the mitigation in a short time. This is judged as inefficient because the EN will continually proceed to a mitigation action.

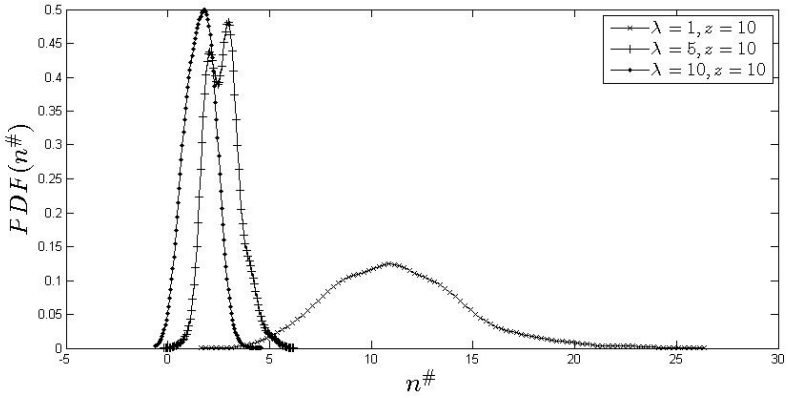

Figure 5: $n^{\#}$ pdf for the DSR mechanism ($z = 10$).

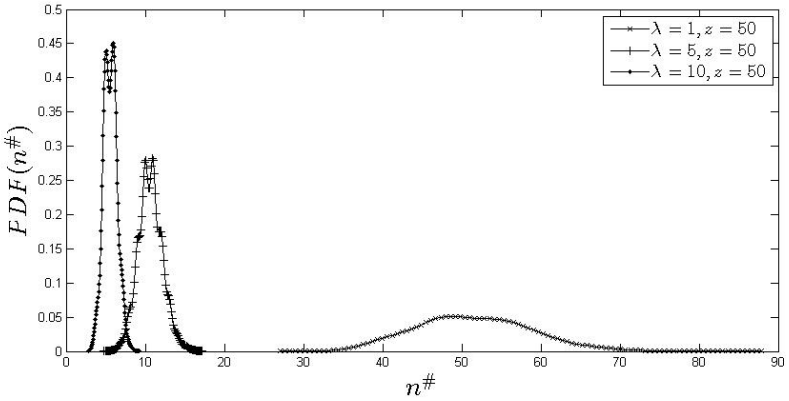

Figure 6: $n^{\#}$ pdf for the DSR mechanism ($z = 50$).

Let us now compare n^* and $n^{\#}$. n^* is seven (7), four and a half (4.5) and four (4) times larger than $n^{\#}$ for $\lambda = \{1, 5, 10\}$, respectively. These results are obtained for $z = 10$. When $z = 50$, n^* is 90%, 72.7% and 66.7% greater than $n^{\#}$ for $\lambda = \{1, 5, 10\}$. The OSR results higher optimal stopping times compared to DSR. The OSR observes QoS_v values and only when there is the need, it decides the mitigation action. It concerns an efficient mechanism as it tries to avoid unnecessary mitigation actions.

In Tables 1, 2 & 3, we provide our comparison results between the proposed OSTM and ESM, MAM and DESM. These results are retrieved for different λ realizations. Recall, that λ affects the 'generation' of QoS violations, i.e., when $\lambda \to 0$, we get low QoS_v values which represent a high QoS and no frequent violations. The opposite, i.e., high QoS_v and frequent violations, stands for a high λ realization. In the aforementioned tables, we can see that the OSTM outperforms the remaining models that are affected by the statistical process adopted to deliver the final result. When $\lambda = 1$, the OSTM does not produce any FP or FN events which means that it is able to detect QOS violations and correctly infer the corresponding mitigation actions. However, the remaining models (ESM, MAM, DESM) also exhibit a good performance even if they produce a few FP and FN events. When we adopt $\lambda = 5$ (the frequency of violations reduces as already explained), we observe that all models produce a higher FN events number compared to the previous experimental scenario which means that they are not able to detect the entire set of violations. Our model results a high ζ compared to the remaining models. The MAM and the DESM are heavily affected by the adopted statistical model that dictates to process multiple historical values before they provide the final result. When $\lambda = 10$, only a few events are realized though our synthetic trace and, again, ESM, MAM & DESM increase the number of FN events leading to a limited performance concerning ζ. The OSTM manages to keep its performance at high levels exhibiting its ability to proactively detect the upcoming violations and 'fire' the relevant mitigation actions.

Table 1: Comparison results for $\lambda = 1$

	OSTM	ESM	MAM	DESM
ϵ	1.000	0.750	0.875	0.417
ζ	1.000	0.750	0.875	0.625
ϕ	1.000	0.996	0.998	0.990

Table 2: Comparison results for $\lambda = 5$

	OSTM	ESM	MAM	DESM
ϵ	1.000	1.000	0.500	0.272
ζ	0.981	0.187	0.188	0.188
ϕ	0.997	0.987	0.984	0.979

Table 3: Comparison results for $\lambda = 10$

	OSTM	ESM	MAM	DESM
ϵ	1.000	1.000	0.475	0.231
ζ	0.980	0.031	0.198	0.063
ϕ	0.998	0.907	0.902	0.890

6. Conclusions and Future Work

In an edge computing setting, end users through their devices as well as applications adopt the desired services to be capable of uploading their data and utilizing edge nodes software solutions. The most critical issue is to secure QoS values at high levels. QoS values define the quality level of the edge nodes and their services and secure the efficient execution of applications. Services monitoring functionalities are necessary in order to identify QoS violations. In this paper, we propose a model that has the responsibility of observing QoS values and progressively deciding whether violations are present. If so, probably the best solution is to select a mitigation action which is represented by data and tasks offloading to the Cloud or to peer nodes. We model the discussed problem and propose a solution adopting the principles of the Optimal Stopping Theory (OST). The proposed mechanism identifies the right time to take a mitigation decision based on sequentially observed values indicating QoS violations. The optimal stopping mechanism can deal with any arbitrary distribution. A set of simulations reveal the strengths of the proposed approach. Our mechanism consists of an efficient solution that outperforms a deterministic model as well as other schemes proposed in the relevant literature. On average, the optimal stopping scheme achieves better results concerning the expected mitigation cost compared to any deterministic model.

Future extensions of our work incorporate the connection of the proposed mechanism with a real monitoring tool. This tool could be responsible for simple as well as more complex services (composition of services). An optimal stopping scheme for manipulating composite services is also in our future research agenda. In this case, the mechanism will receive multiple QoS values and should decide when it is the right time to decide the mitigation for the entire set of services (the composite service) or for a subset of them.

References

[1] Abderrahim, M., Ouzzif, M., Guillouard, K., Francois, J., Lebre, A., 'A Holistic Monitoring Service for Fog/Edge Infrastructures: a Foresight Study', The IEEE 5th International Conference on Future Internet of Things and Cloud, 2017, pp. 3337–344.

[2] Aceto, G., Botta, A., De Donato, W., Pescape, A., 'Cloud Monitoring: A Survey', Journal of Computer Networks, vol. 57(9), 2013, pp. 2093–2115.

[3] Ahmed, A., Ahmed, E., 'A survey on mobile edge computing', in 2016 10th International Conference on Intelligent Systems and Control (ISCO), 2016, pp. 1-8.

[4] Al-Hazmi, Y., Campowsky, K., Magedanz, T., 'A monitoring system for federated clouds', In Proceedings of the 1st IEEE International Conference on Cloud Networking, 2012, pp. 68-74.

[5] Alrifai, M., Risse, T., Dolog, P., Nejdl, W., 'A scalable approach for QoS-based web service selection', in Proceedings of Service-Oriented Computing-ICSOC Workshops, 2009.

[6] Armstrong, D. J., 'Enhancing Quality of Service in CC Through Novel Resource Management', PhD Thesis, University of Leeds, 2012.

[7] Atzori, L., Iera, A., Morabito, G., 'The internet of things: A survey', Computer networks, vol. 54, no. 15, pp. 2787-2805, 2010.

[8] Bartalos, P., Bielikov, M., 'Automatic dynamic web service composition: A survey and problem formalization', Computing and Informatics, vol. 30, no. 4, pp. 793-827, 2012.

[9] Bradshaw, R., Desai, N., Freeman, T., Keahey, K., 'A scalable approach to deploying and managing appliances', in Proceedings of the TerraGrid Conference, 2007.

[10] Brandn, A., Prez, M. S., Montes, J., Sanchez, A., 'FMonE: A Flexible Monitoring Solution at the Edge', Wireless Communications and Mobile Computing, vol. 2018, art. 2068278, 2018.

[11] Caglar, F., Gokhale, A., 'iOverbook: intelligent resource-overbooking to support soft real-time applications in the cloud', In Proceedings of the 7th IEEE International Conference on Cloud computing (CLOUD) 2014, pp. 538-545.

[12] Chen, Z., Hu, W., Wang, J., Zhao, S., Amos, B., Wu, G., Ha, K., Elgazzar, K., Pillai, P., Klatzky, R., Siewiorek, D., Satyanarayanan, M., 'An Empirical Study of Latency in an Emerging Class of Edge Computing Applications for Wearable Cognitive Assistance', in Proc. of the 2nd ACM/IEEE Symposium on Edge Computing, 2017

[13] Chieu, T., Mohindra, A., Karve, A., Segal, A., 'Solution based deployment of complex application services on a cloud', in Proceedings of the 2010 IEEE International Conference on Service Operations and Logistics and Informatics, 2010.

[14] Clayman, S., Galis, A., Mamatas, L., 'Monitoring virtual networks with lattice', In Proceedings of 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, 2010, pp. 239-246.

[15] Dastjerdi, A. V., Buyya, R., 'Fog Computing: Helping the Internet of Things Realize Its Potential', The Computer Journal, vol. 49(8), 2016, pp. 112-116.

[16] Dastjerdi, A. V., Buyya, R., 'Compatibility-aware Cloud Service Composition Under Fuzzy Preferences', IEEE Transactions on Cloud Computing, vol. 2(1), 2014, pp. 1–13.

[17] Dastjerdi, A. V., Tabatabaei, S. G. H., Buyya, R., 'A Dependency Aware Ontology Based Approach for Deploying Service Level Agreement Monitoring Services in Cloud', Software Practice and Experience, vol. 42, pp. 501, 518, 2012.

[18] Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Mhr, T., Loichate, M., 'Building a mosaic of clouds', in Proceedings of EuroPar 2010 Parallel Processing Workshops, Springer, 2011.

[19] Dusia, A. , Yang, Y. , Taufer, M., 'Network quality of service in Docker containers', In Proceedings of 2015 IEEE International Conference on Cluster Computing (CLUSTER), 2015, pp. 527-528.

[20] Evans, K., Jones, A., Preece, A., Quevedo, F., Rogers, D., Spasic, I., Taylor, I., Stankovski, V., Taherizadeh, S., Trnkoczy, J., Suciu, G., Suciu, V., Martin, P., Wang, J., Zhao, Z., 'Dynamically reconfigurable workflows for time-critical applications', In Proceedings of International workshop on Workflows in support of large-scale science, 2015, pp. 1-10.

[21] Farokhi, S., Lakew, E.B., Klein, C., Brandic, I., Elmroth, E., 'Coordinating CPU and memory elasticity controllers to meet service response time constraints', In Proceedings of International Conference on Cloud and Autonomic Computing, 2015, pp. 69-80.

[22] Goiri, I., Julia, F., Fito, J. O., Macias, M., Guitart, J., 'Resource-Level QoS Metric for CPU-based Guarantees in Cloud Providers', in Proceedings of the 7th international conference on Economics of grids, clouds, systems, and services, 2010.

[23] Grobmann, M., Klug, C., 'Monitoring Container Services at the Network

Edge', in Proceedings of the 29th International Teletraffic Congress, ITC, 2017, pp. 130-133.

[24] Huckova, I., Cicak, P., 'Advanced Network Monitoring using the Selection of Critical Nodes', in 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019.

[25] Hussain, W., Sohaib, O., 'Analysing Cloud QoS Prediction Approaches and Its Control Parameters: Considering Overall Accuracy and Freshness of a Dataset', IEEE Access, vol. 7, 2019, pp. 82649–82671.

[26] Islam, S., Keung, J., Lee, K., Liu, A., 'Empirical prediction models for adaptive resource provisioning in the cloud', Future Generation Computer Systems, vol. 28(1), 2012, pp. 155-162.

[27] Jamshidi, P., Sharifloo, A.M., Pahl, C., Metzger, A., Estrada, G., 'Self-learning cloud controllers: fuzzy q-learning for knowledge evolution', In Proceedings of International Conference on Cloud and Autonomic Computing, 2015, pp. 208-211.

[28] Karlapudi, H., Martin, J., 'Web application performance prediction', In Proceedings of the IASTED International Conference on Communication and Computer Networks, 2004.

[29] Kolomvatsos, K., Anagnostopoulos, C., 'Multi-criteria Optimal Task Allocation at the Edge', Elsevier Future Generation Computer Systems, vol. 93, 2019, pp. 358–372.

[30] Konstantinou, A. V., Eilam, T., Kalantar, M., Totok, A. A., Arnold, W., Snible, E., 'An architecture for virtual solution composition and deployment in infrastructure clouds', in Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing, 2009.

[31] Kreutz, D., Ramos, F., Esteves, P., Esteve Rothenberg, C., Azodolmolky, S., Uhlig, S., 'Software-defined networking: A comprehensive survey', Proceedings of the IEEE, vol. 103(1), pp. 14-76, 2015.

[32] Kwon, S., Noh, J., 'Implementation of monitoring system for cloud computing', IJMER, vol. 3(4), 2013, pp. 1916-1918.

[33] Lecue, F., Mehandjiev, N., 'Towards scalability of quality driven semantic web service composition', in Proceedings of IEEE International Conference on Web Services, 2009.

[34] Leitner, P., Inzinger, C., Hummer, W., Satzger, B., Dustdar, S., 'Application-level performance monitoring of cloud services based on the complex event processing paradigm', In Proceedings of the 5th IEEE International Conference on Service-Oriented Computing and Applications, 2015,pp. 1-8.

[35] Li, Q., Hao, Q., Xiao, L., Li, Z., 'Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control', in 1st International Conference on Information Science and Engineering, 2010, pp. 99–102.

[36] Li, J., Qiu, M., Niu, J. W., Chen, Y., Ming, Z., 'Adaptive Resource Allocation for Preemptable Jobs in Cloud Systems', in 10th International Conference on Intelligent System Design and Application, 2011, pp. 31–36.

[37] Lu, J., Wang, J., 'Performance modeling and analysis of Web Switch', In Proceedings of the 31st Annual International Conference on Computer Measurement (CMG05), Orlando, 2005.

[38] Manvi, S. S., Shyam, G. K., 'Resource Management for Infrastructure as a Service (IaaS) in Cloud Computing: A Survey', Journal of Networks and Computer Applications, vol. 41, pp. 424–440, 2014.

[39] Mastelic, T. , Emeakaroha, V.C. , Maurer, M. , Brandic, I., 'M4Cloud: generic application level monitoring for resource-shared cloud environments', In Proceedings of the 2nd International Conference on Cloud Computing and Services Science, 2012, pp. 522-532.

[40] Meera, A., Swamynathan, S.,'Agent based resource monitoring system in IaaS cloud environment', In Proceedings of the 1st International Confer-

ence on Computational Intelligence: Modeling Techniques and Applications, 2013, pp. 200-207.

[41] Mei, R. D., Meeuwissen, H. B., Phillipson, F., 'User perceived Quality-of-Service for voice-over-IP in a heterogeneous multi-domain network environment', In Proceedings of ICWS, 2006.

[42] Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., De Turck, F., Boutaba, R., 'Network function virtualization: State-of-the-art and research challenges', IEEE Communications Surveys & Tutorials, vol. 18(1), pp. 236-262, 2015.

[43] OpenFog Consortium, 'OpenFog Reference Architecture for Fog Computing', Report, 2017, retrieved at August 2019 from https:www.iiconsortium.orgpdfOpenFog_Reference_Architecture_2_09_17.pdf.

[44] Peskir, G. and Shiryaev, A., 'Optimal stopping and free boundary problems', ETH Zuerich, Birkhauser, 2006.

[45] Pham, T., Truong, H., Dustdar, S., 'Elastic high performance applicationsa composition framework', in Proceedings of IEEE Asia-Pacific Services Computing Conference, 2011.

[46] Preeth, E.N., Mulerickal, F.J.P., Paul, B., Sastri, Y., 'Evaluation of Docker containers based on hardware utilization', In Proceedings of 2015 International Conference on Control Communication & Computing India (ICCC), 2015, pp. 697-700.

[47] Rodriguez, A., Carretero, J., Bergua, B., Garcia, F., 'Resource selection for fast large-scale virtual appliances propagation', in Proceedings of the IEEE Symposium on Computers and Communications, 2009.

[48] Rosenberg, F., Muller, M., Leitner, P., Michlmayr, A., Bouguettaya, A., Dustdar, S., 'Metaheuristic optimization of large scale qos-aware service compositions', in Proceedings of IEEE International Conference on Services Computing, 2010.

[49] Rossi, F., Oliveira, I., de-Rose, C., Calheiros, R., Buyya, R., 'Non-invasive estimation of cloud applications performance via hypervisors operating systems counters', In Proceedings of the 14th International Conference on Networks, 2015, pp. 177-184.

[50] Seo, K., Hwang, H., Moon, I., Kwon, O., Kim, B., 'Performance comparison analysis of Linux container and virtual machine for building cloud', Adv. Sci. Technol. Lett. 66, 2014, pp. 105-111.

[51] Stankovski, V., Trnkoczy, J., Taherizadeh, S., Cigale, M., 'Implementing time-critical functionalities with a distributed adaptive container architecture', In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services (iiWAS2016), ACM, Singapore, 2016, pp. 455-459.

[52] Taherizadeh, S., Jones, A., Taylor, I., Zhao, Z., Stankovski, V., 'Monitoring Self-Adaptive Applications within Edge Computing Frameworks: A State-of-the-art Review', The Journal of Systems and Software, vol. 136, 2018, pp. 19–38.

[53] Wamser, F., Loh, F., Seufert, M., Tran-Gia, P., Bruschi, R., Lago, P., 'Dynamic cloud service placement for live video streaming with a remote-controlled drone', In Proceedings of the 15th IFIP/IEEE International Symposium on Integrated Network Management (IM) (Demonstration), 2017.

[54] Wang, N., Varghese, B., Matthaiou, M., Nikolopoulos, D., 'ENORM: A Framework For Edge Node Resource Management', IEEE Transactions on Services Computing, 2017.

[55] Wu, N., Tang, A., 'End-to-end Network Throughput Optimization Through Last-mile Diversity', in 52nd Annual Conference on Information Sciences and Systems (CISS), 2018.

[56] Wu, D., Liu, S., Zhang , L., Terpennya, J., Gaoc, R., Kurfessd, T., Guzzob, J., 'A fog computing-based framework for process monitoring and prognosis

in cyber-manufacturing', Journal of Manufacturing Systems, vol. 43, 2017, pp. 25-34.

[57] Xiong, K., Perros, H., 'Service Performance and Analysis in Cloud Computing', in World Conference on Services, 2009.

[58] Xiong, P., Pu, C., Zhu, X., Griffith, R., 'vPerfGuard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments', In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, 2013, pp. 271-282.

[59] Yao, Y., Chen, H., Qos-aware service composition using nsga-ii, in Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, 2009.

[60] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., Jue, J., 'All One Needs to Know about Fog Computing and Related Edge Computing Paradigms', Journal of Systems Architecture, vol. 98, 2019, pp. 289–330.

[61] Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., Yang, X., 'A Survey on the Edge Computing for the Internet of Things', IEEE Access, vol. 6, 2017.

[62] Zhang, M., Ranjan, R., Nepal, S., Menzel, M., Haller, A., A Declarative Recommender System for Cloud Infrastructure Services Selection, Economics of Grids, Clouds, Systems and Services, Lecture Notes in Computer Science, Springer, 2012.