An Open-Source Experimentation Framework for the Edge Cloud Continuum

Georgios Koukis[‡†], Sotiris Skaperas[*†], Ioanna Angeliki Kapetanidou[‡†], Vassilis Tsaoussidis[‡†], Lefteris Mamatas[*†]
[†] Athena Research and Innovation Center, Greece
[‡] Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece
[*] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
Emails: {gkoukis, ikapetan, vtsaousi}@ee.duth.gr, {sotskap, emamatas}@uom.edu.gr

***Abstract*—The CODECO Experimentation Framework is an open-source solution designed for the rapid experimentation of Kubernetes-based edge cloud deployments. It adopts a microservice-based architecture and introduces innovative abstractions for (i) the holistic deployment of Kubernetes clusters and associated applications, starting from the VM allocation level; (ii) declarative cross-layer experiment configuration; and (iii) automation features covering the entire experimental process, from the configuration up to the results visualization. We present proof-of-concept results that demonstrate the above capabilities in three distinct contexts: (i) a comparative evaluation of various network fabrics across different edge-oriented Kubernetes distributions; (ii) the automated deployment of EdgeNet, which is a complex edge cloud orchestration system; and (iii) an assessment of anomaly detection (AD) workflows tailored for edge environments.**

***Index Terms*—Testbed-based experimentation, Edge Computing, Kubernetes, Edge Cloud Networking, Network Plugins, Anomaly Detection**

I. Introduction

Edge computing has become a paramount paradigm to meet the requirements of the ever-increasing Internet of Things (IoT) applications and services that often require real-time processing. By bringing computation and storage closer to the data sources and end-users, edge computing contributes to reduced processing and response times, network load, and overall improved delivery and system performance.

Nevertheless, edge devices may be resource-constrained in terms of computational power, memory and energy support. Therefore, edge solutions should be designed in a resource-efficient manner to achieve optimal performance. In this context, microservice-based solutions have gained significant popularity as, unlike traditional monolithic applications, they allow for offloading parts of the service to edge devices [1], leading to improved resource efficiency. In particular, microservice-based applications consist of several loosely-coupled microservices, each being responsible for a specific, single-purpose part of the application's functionality. These microservices can be deployed, scaled, and updated independently, offering distinct advantages such as composable software design, programming heterogeneity, and simplified debugging [2], [3].

Kubernetes (K8s), the de facto solution for service orchestration, facilitates automated service deployment and configuration, while also providing advanced scheduling capabilities [4], [5]. However, being originally designed for traditional cloud environments, K8s might not be well-suited to support today's microservice-based, time-critical applications in the edge [6]. In this light, several lightweight Kubernetes derivatives (e.g., K0s[1], K3s[2], MicroK8s[3]) have been developed specifically for resource-constrained or low-footprint edge devices, aiming to better address the unique requirements of the edge [7]. Additionally, various platforms, both K8s and non-K8s focused have been proposed for the edge, such as KubeEdge [8], KubeOne[4], and Open Horizon [9], each one focusing on particular aspects, e.g., resource consumption, device computational power, etc. The optimal K8s distribution for each case depends on the deployment scenario and the associated key driving factors.

Besides the selected K8s distribution, there is a growing need for intelligent, flexible and holistic manipulation of edge compute and network resources. Motivated by this, the CODECO project [10] adopts a cross-layer adaptation approach, aiming to ensure enhanced performance for microservice-based applications across the entire Edge-Cloud continuum. CODECO introduces novel components, including ACM for automated configuration, deployment and monitoring of edge cloud resources, MDM for data workflow observability, SWN for scheduling and re-scheduling of application workloads, PDLC taking intelligent decisions for edge cloud orchestration, and NetMA providing network-awareness to CODECO and handling secure connectivity across pods.

From this point of view, an edge cloud continuum experimentation framework should be able to incorporate holistic experimental definitions for all layers of the protocol stack and main K8s features, including the support of alternative edge infrastructures (e.g., choice of hypervisor and server hardware) and virtualization technologies, to accommodate the diversity of edge cloud demands [11]. Other important requirements include: i) appropriately designed abstractions to reduce experimentation complexity, ii) advanced automation and reliability, enabling the execution of experiments with

[1]https://k0sproject.io/
[2]https://www.rancher.com/products/k3s
[3]https://microk8s.io/
[4]https://www.kubermatic.com/products/kubermatic-kubeone/edge/

different combinations of cross-layer parameters, that may also serve as reference points to evaluate a diverse set of intelligent algorithms, spanning from machine learning (ML) to optimization techniques, and, iii) abstractions apt to support modular deployment extensions.

Existing open testbed solutions, incorporate cutting-edge technologies, such as 5G/6G, distributed/cloud computing, ML/AI and optical technologies, that enable a wide range of experimentation activities [12]. Foundational paradigms of such testbeds include the EU-located SLICES testbed [13] and the US-located FABRIC [14], both recognized for their large-scale capabilities. A similar approach is adopted by PlanetLab [15] where users and institutes can contribute nodes, focusing on the federation of the provided heterogeneous infrastructures. An evolution of Planetlab is EdgeNet [16], which targets globally distributed edge-cloud environments. EdgeNet utilizes native K8s for the deployment and node contribution processes, extending the latter with custom resources (CRs).

Another K8s-centered solution is our recently introduced ClusterSlice [17] platform, which is a zero-touch solution for transforming testbed resources into fully operational K8s slices. The CODECO Experimentation Framework extends the novel Resource and Infrastructure Manager operators of ClusterSlice to be used as non-K8s components. These two abstractions handle the manipulation of VMs and compute resources, respectively. Compared to ClusterSlice, it is lightweight (i.e., requires only docker for execution) as well as it supports additional edge capabilities and environments (e.g., EdgeNet) and experimentation automation features (i.e., through its own innovative abstractions).

CODECO Experimentation Framework is based on a microservice-based architecture that offers:

1) holistic experiment configuration, including the deployed infrastructure and K8s configuration parameters as well as the input of the experiments to be conducted,
2) innovative, modular and extensive abstractions, to accommodate a diverse set of experiments based on Ansible playbook templates,
3) complete experimentation automation, from cluster deployment to experiment execution and results processing.

Along these lines, the framework possesses additional advantages: i) reproducible experimentation with one-liner commands and versioning of components, ii) minimal operational support and risk since the deployments can be easily deleted and then re-configured, iii) integration with external edge environments and testbeds, e.g., EdgeNet, Cloudlab etc. The novel CODECO Experimentation Framework's capabilities are demonstrated through three proof-of-concept experiments, ranging from evaluating different network plugins across various K8s distributions to deploying EdgeNet software and assessing various anomaly detection (AD) approaches.

The remainder of the paper is structured as follows. In Section II, we detail the architecture of CODECO Experimentation Framework along with its individual components. In Section III, we give our proof-of-concept results, highlighting the main features and capabilities of the framework. Finally, Section IV outlines our conclusions and future plans.

```
# generic configuration
experiment_title="my_experiment"
username=athena
password=<ENCODED_PASSWORD>
# infrastructure configuration
infra_manager_ip=83.212.134.23
infra_manager_name=codecocloud
infra_manager_type=xcp-ng
use_snapshots=true
node_osimage=ubuntu-22-clean
master_hosts=["athm1"]
master_ips=["83.212.134.27"]
master_macs=["66:16:91:ec:09:03"]
worker_hosts=["athw1"]
worker_ips=["83.212.134.35"]
worker_macs=["66:16:91:ec:09:11"]
# kubernetes configuration
k8s_type=vanilla
k8s_scheduler=swm
k8s_networkfabric=l2s-m
k8s_version="1.23"
# application configuration
app_names=["docker", "dashboard"]
app_scopes=["all", "cluster"]
# experiment definition
exp_manager=cni-plugins
exp_input=["k8s-flannel", "k3s-calico"]
exp_metrics=["latency"]
replications_number=10
results_output="PDF"
```

Fig. 1: A Simple Experiment Definition File

II. Design and Implementation

Here, we present the CODECO Experimentation Framework. Specifically, we describe a simple experiment definition file and give an overview of the framework's architecture.

A. Experiment Descriptor

The framework encompasses the entire experimentation process, beginning with the declarative definition of experiments for cluster development and the intended experiments. It then proceeds to the automated deployment of clusters and applications, the execution of experiments based on defined metrics, and ultimately, to the results' acquisition and processing.

Fig. 1 gives an example experiment descriptor. It includes parameters for the infrastructure configuration, considering cluster information such as cluster name, IP and hypervisor, OS image, IP and MAC addresses for each cluster node (master or worker). In addition, specific K8s configuration settings can be defined, such as K8s flavor and version, scheduler or networking plugin; applications to be deployed, such as monitoring tools, Docker, and multi-cluster solutions; or experiment parameters, including metrics, experiment controller to support (i.e., container identifying the particular experiment), number of replications, and output file creation.

B. Architecture

The architecture of CODECO Experimentation Framework comprises the following components, as shown in Fig. 2. All components take the form of microservices (i.e., containers) and can be independently extended to accommodate additional technologies or features.

The *Experiment Manager* oversees and coordinates all experiment processes, maintaining smooth execution without unexpected issues, such as ensuring reproducibility and facilitating automatic deployment and control of experiments.

The *Infrastructure Manager* realizes a technology-agnostic abstraction over heterogeneous test-beds and cloud systems, which is responsible to allocate the cluster nodes from physical node or VM allocation to OS installation phases. In particular, specifications related to the allocation of nodes (e.g., number of master/worker nodes, OS image and snapshot usage) are the inputs of Infrastructure Manager, which in turn returns the addresses of allocated physical nodes or VMs. After the allocation of cluster nodes, a number of *Resource Managers* are being deployed, one for each node.

The *Resource Managers* provide a node-level automation abstraction for the software deployment of cluster nodes. Specifically, they oversee the deployment of user-defined applications, supporting a range of them through the utilization of Ansible playbook templates (e.g., Docker, K8s dashboard, Liqo, Submariner, Argo Workflows, etc.), while also determining their versioning and deployment scope (e.g., deployment at the cluster level, across all servers etc.).

The component responsible for executing specific experiments once the cluster is operational is the *Experiment Controller*. It offers a straightforward and automated experimentation approach, facilitating -among others- the performance evaluation of various CNI plugins and AD methods. The user is only required to input the experiment definition with the name of particular *Experiment Controller* to use, specific metrics, along with the desired number of replications. Subsequently, the framework executes the experiments and produces the corresponding results.

Finally, the *Results Processor* evaluates and post-processes the results of the experiments. Its tasks include performing statistical evaluations (e.g., mean values assessment), plotting the results based on the pre-defined metrics, and automating the creation of LaTeX-based report PDF files, incorporating the plots generated from the experiments.

III. Proof-of-concept Results

This section details our experimentation setup and considered scenarios. In this regard, our proof-of-concept results demonstrate three distinct cases CODECO Experimentation Framework is utilized. In particular, we examine the: i) performance of network plugins across standard and Edge-oriented K8s distributions, ii) automatic deployment of a complicated edge system, i.e., EdgeNet, and iii) implementation of AD methods as on-demand K8s workflows.

A. Experimentation Setup

We execute the experiments in two testbeds referred to as ATH and UOM, which are located at the ATHENA Research Center and the University of Macedonia, respectively. Both testbeds use the XCP-ng virtualization platform[5]. The former consists of a single Dell PowerEdge T640 physical machine equipped with an Intel(R) Xeon(R) Silver 4210R processor running at 2.40GHz, a 16-core CPU and 64GB RAM. The latter testbed comprises two Dell PowerEdge R630 physical servers, each one with 2 Intel(R) Xeon(R) CPU E5-2620 v4 processors operating at 2.10GHz with 16-core CPUs.

B. Experimentation Scenarios

1) Edge-oriented K8s Distributions and Network Plugins: The CODECO Experimentation Framework supports a variety of K8s distributions and network plugins ranging from resource-intensive, production-based and feature-rich solutions to more lightweight and resource-constrained approaches.

In particular, the framework supports four widely used K8s distributions, i.e., vanilla K8s, K3s, K0s and MicroK8s. The feature-rich vanilla K8s is the standard K8s flavor, while the rest are lightweight distributions targeting resource-constrained environments, like the edge. Furthermore, we support EdgeNet, which is a cloud orchestration system for the edge cloud continuum.

Similarly, various network plugins can be utilized for each K8s distribution: (i) vanilla K8s: Flannel, Multus, Calico, WeaveNet, Cilium, Kube-Router, Kube-OVN, Antrea; (ii) K3s: Flannel, Calico, Cilium; (iii) K0s: Kube-Router, Calico; (iv) Microk8s: Calico, Flannel, Kube-OVN. In addition to the above CNI plugins, we have also implemented the support of L2S-M solution [18], i.e., utilized by CODECO NetMA, within vanilla K8s (referred to as $k8s - l2sm - v1$). The considered network infrastructure (e.g., underlay or overlay) and the provided features (e.g., advanced security/encryption, programmability, network policies, etc.) vary among the plugins. Notably, Flannel, WeaveNet and Kube-Router provide simpler and more lightweight solutions; Calico, Cilium, Antrea and Kube-OVN support advanced and programmable features, while Multus allows for the simultaneous utilization of multiple plugins within a cluster.

Figs. 3 and 4 illustrate the resource consumption (CPU, RAM) and throughput performance of the supported network solutions across the different K8s distributions. The former focuses on the standard K8s flavor, while the latter evaluates the performance of networking solutions deployed in edge-related distributions, i.e., K3s, K0s, MicroK8s and EdgeNet. In this context, various communication conditions, i.e., "Idle", "Pod-to-Pod" (p2p), and "Pod-to-Service" (p2s), are evaluated both for TCP and UDP transport layer protocols. The Kubernetes Network Benchmark (knb) tool[6] is utilized to obtain the aforementioned metrics in an intra-host deployment in the ATH testbed. The deployment of the distributions and

[5]https://xcp-ng.org

[6]https://github.com/InfraBuilder/k8s-bench-suite

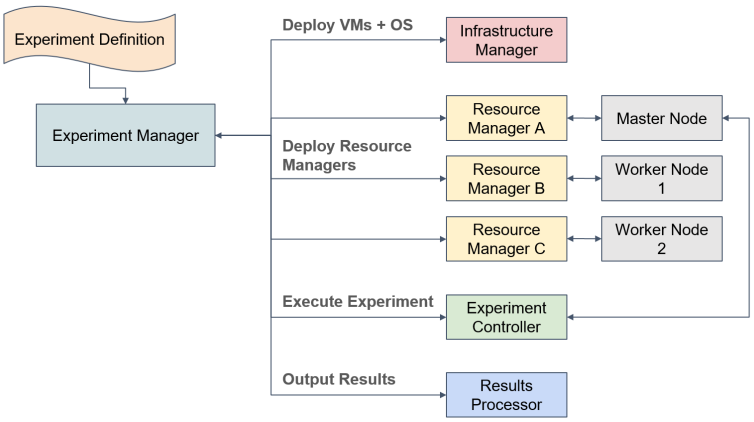
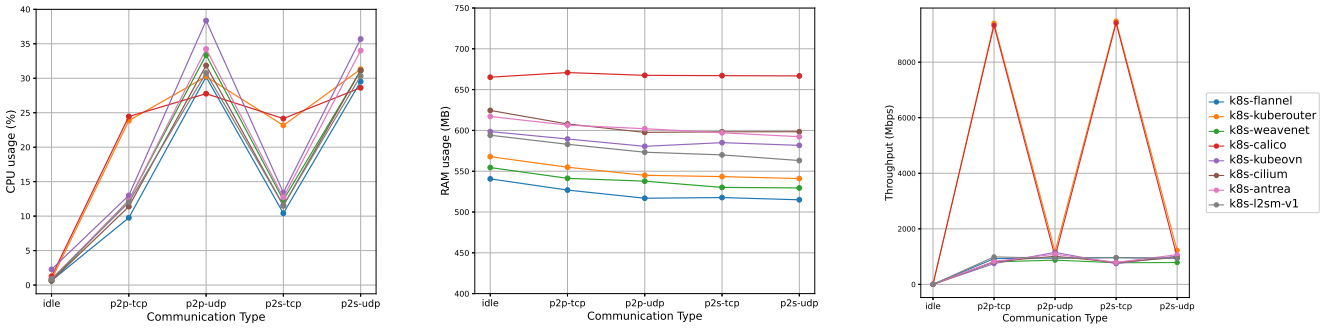


Fig. 2: Main architectural components and interactions.



(a) CPU utilization (b) RAM consumption (c) Achieved throughput

Fig. 3: CPU, RAM usage and throughput per CNI plugin, for the K8s distribution.

respective plugins is carried out automatically by the framework resulting in identical clusters which comprise one master and two worker nodes (VMs). In [19], we document a relevant comprehensive performance evaluation. Consequently, our framework is capable of experimenting with multiple networking solutions over various edge-oriented K8s flavors.

2) EdgeNet Deployment: Here, we demonstrate the capability of CODECO Experimentation Framework to handle complex edge system deployments, such as the installation of EdgeNet, an open-source solution designed to extend K8s to the Edge [20], [21]. Such process involves installing EdgeNet prerequisites, e.g., creates kubeconfig and ssh-key secret certificates, configures wireguard, creates CRs for VPN peering, and installs the EdgeNet features implemented as CR Definitions and operators. Such deployment required also improvements in the EdgeNet software itself, e.g., to improve its autonomic deployment, compatibility and versioning.

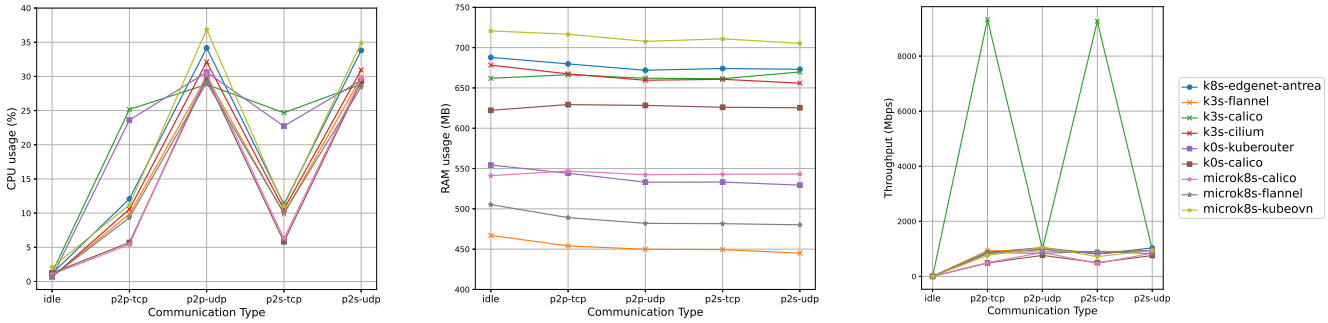
After the installation, a functional CODECO-EdgeNet K8s cluster is deployed, with supported functionalities, such as: i) *Multi-tenancy*: role-based approach in the shared cluster, e.g., create and accept tenant and role requests, create slices, subnamespaces, allocate resource quotas, etc.; and ii) *Multi-provider*: users from around the world can contribute nodes (VMs or physical ones) to the cluster, with a single command.

Fig 5 illustrates the output of some basic $kubectl$ commands, showcasing the successful installation of EdgeNet software, the establishment of the cluster, and the node contribution processes. In particular, a modified version of the Definition File in II-A is used for the installation of EdgeNet utilizing the specific EdgeNet-related automation features, followed by distributed nodes (VMs) that join this cluster with one-liner commands. Subsequently, we complement Subsection III-B1 by evaluating the performance of the Antrea CNI plugin within this local EdgeNet cluster (referred to as $k8s - edgenet - antrea$) in Fig 3.

3) Anomaly Detection Workflows for the Edge: Lastly, we demonstrate CODECO's Experimentation Framework capabilities to assess the impact of ML algorithms in edge cloud environments, in terms of actual response time and resource utilization. Here, we focus on the evaluation of the following 5 AD detectors: i) typical CUSUM detector (tCUSUM) [22], ii) ratio-type CUSUM (r-t CUSUM) [22], iii) ARMA-based CUSUM (pCUSUM) [23], iv) restarted Bayesian (rBOCP) [24], and, iv) off-line CUSUM [25]. Note that the considered detectors operate online and are applied on the UOM testbed.

The AD methods are implemented as K8s argo workflows[7]

[7]https://github.com/argoproj/argo-workflows



(a) CPU utilization (b) RAM consumption (c) Achieved throughput

Fig. 4: CPU, RAM usage and throughput per CNI plugin, for lightweight distributions.

```
athena@athm1:~$ kubectl get nodes -o wide
 NAME                     STATUS   ROLES           AGE    VERSION    INTERNAL-IP       OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
 athm1                    Ready    control-plane,  28h    v1.23.17   83.212.134.25     Ubuntu 22.04.2 LTS   5.15.0-71-generic   containerd://1.6.24
                                   master
 athw2                    Ready    <none>          28h    v1.23.17   83.212.134.29     Ubuntu 22.04.2 LTS   5.15.0-71-generic   containerd://1.6.24
 athw3                    Ready    <none>          28h    v1.23.17   83.212.134.30     Ubuntu 22.04.2 LTS   5.15.0-71-generic   containerd://1.6.24
 gr-a-7230.edge-net.io    Ready    <none>          15m    v1.23.17   83.212.134.27     Ubuntu 22.04.2 LTS   5.15.0-86-generic   containerd://1.5.11
 gr-b-89c3.edge-net.io    Ready    <none>          108s   v1.23.17   195.251.209.229   Ubuntu 22.04.2 LTS   5.15.0-71-generic   containerd://1.5.11
 gr-b-abf4.edge-net.io    Ready    <none>          3m26s  v1.23.17   195.251.209.231   Ubuntu 22.04.2 LTS   5.15.0-71-generic   containerd://1.5.11

athena@athm1:~$ kubectl get nodecontributions
 NAME        ADDRESS           PORT   ENABLED   STATUS          AGE
 gr-a-7230   83.212.134.27     22     true      Node Accessed   16m
 gr-b-89c3   195.251.209.229   22     true      Node Accessed   2m34s
 gr-b-abf4   195.251.209.231   22     true      Node Accessed   4m13s
```

Fig. 5: EdgeNet installation and node contribution output commands.

(operated/removed on demand) and use a client-server application. In detail, the experimental parameters are defined through the workflows, e.g., the type of the AD method and the number of simultaneous clients to be deployed. The client-server communication is achieved through a REST API. The client transmits, in real-time, new data samples to the server, which employs the AD mechanisms and alerts the clients for anomalies. The servers also monitor the resource consumption of the AD mechanisms using an extension of knb tool.

Fig. 6 illustrates the performance metrics for the considered AD approaches, assessed over 100 Monte Carlo simulations, for one client transmitting synthetic data following a piecewise Gaussian distribution with one anomaly in the mean value. Particularly, Fig. 6a depicts the common detection gap (in data points t) between the occurrence and the estimation of an anomaly. In addition, the CODECO Experimentation Framework extends the evaluation metrics to the actual detection gap and response time along with the CPU and memory consumption for each AD method, as shown in Figs. 6b – e. This provides valuable insights for practitioners about the performance of AD methods in real-world edge applications that cannot be revealed through typical statistical metrics. For instance, when comparing rBOCP with the online CUSUM procedures, the first concludes on a substantially higher actual detection gap in msec, despite providing the shortest detection gap in data points. A thorough evaluation of different AD methods in edge-cloud systems can be found in our previous work [26].

IV. Conclusions and Future Work

This paper introduced the open-source CODECO Experimentation Framework[8] tailored for experimenting in the Edge. We presented its architecture, underscoring its capacity to facilitate the deployment of clusters, applications and experiments in a holistic and automated manner. Additionally, we demonstrated the framework's significance through three proof-of-concept examples. Our future plans involve the support of additional pluggable K8s features and applications (e.g., the CODECO components), the investigation of multi-cluster solutions and its interconnection with external testbeds.

Acknowledgment

This work is supported by the Horizon Europe CODECO Project under Grant number 101092696.

[8]We released the CODECO Experimentation Framework and our port of EdgeNet in https://gitlab.eclipse.org/eclipse-research-labs/codeco-project/experimentation-framework-and-demonstrations

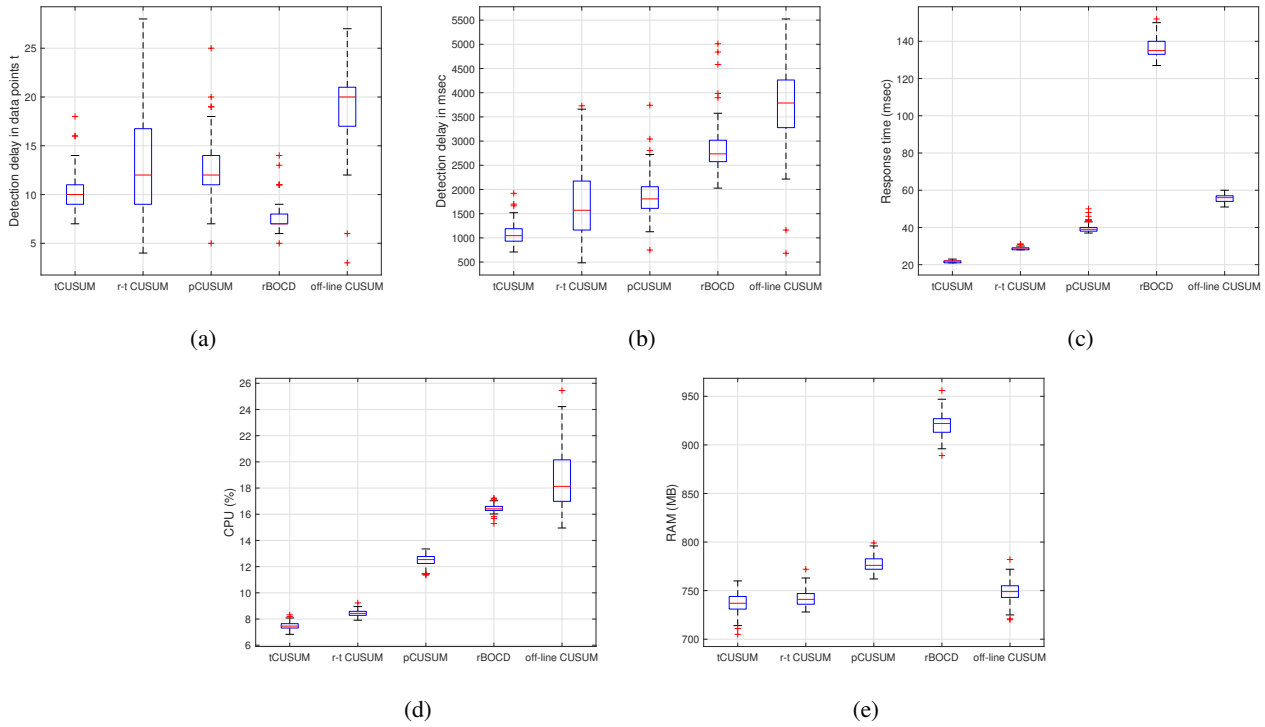


Fig. 6: Comparison of (a) detection gap in data points t, (b) detection gap in msec, (c) response time in msec, (d) CPU, and, (e) RAM consumption between the anomaly detection methods.

REFERENCES

[1] K. Fu, W. Zhang, Q. Chen *et al.*, "Adaptive resource efficient microservice deployment in cloud-edge continuum," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1825–1840, 2021.

[2] Y. Gan, Y. Zhang, D. Cheng, Shetty *et al.*, "An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2019, pp. 3–18.

[3] H. Zhao, S. Deng, Z. Liu *et al.*, "Distributed redundant placement for microservice-based applications at the edge," *IEEE Trans. on Services Comput.*, vol. 15, no. 3, pp. 1732–1745, 2022.

[4] C. Carrión, "Kubernetes scheduling: Taxonomy, ongoing issues and challenges," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–37, 2022.

[5] Y. Han, S. Shen, X. Wang *et al.*, "Tailored learning-based scheduling for kubernetes-oriented edge-cloud system," in *IEEE INFOCOM 2021 - IEEE Conf. on Comput. Commun.*, 2021, pp. 1–10.

[6] S. Böhm and G. Wirtz, "Profiling lightweight container platforms: Microk8s and K3s in Comparison to Kubernetes," in *Proc. ZEUS*, Bamberg, Germany, 2021, vol. 2389, pp. 65–73.

[7] H. Koziolek and N. Eskandani, "Lightweight kubernetes distributions: A performance comparison of microk8s, k3s, k0s, and microshift," in *Proc. ACM/SPEC Int. Conf. Perform. Eng.*, 2023, pp. 17–29.

[8] Y. Xiong, Y. Sun, L. Xing *et al.*, "Extend cloud to edge with kubeedge," in *2018 IEEE/ACM Symposium on Edge Comput. (SEC)*, 2018, pp. 373–377.

[9] "LF EDGE project," https://www.lfedge.org/projects/openhorizon/, Accessed, December 2023.

[10] "CODECO Horizon Europe project," https://he-codeco.eu/, Accessed, December 2023.

[11] P. Valsamas, S. Skaperas, L. Mamatas *et al.*, "Virtualization Technology Blending for resource-efficient edge clouds," *Computer Netw.*, vol. 225, p. 109646, 2023.

[12] G. Z. Papadopoulos, A. Gallais, G. Schreiner *et al.*, "Thorough IoT testbed characterization: From proof-of-concept to repeatable experimentations," *Computer Netw.*, vol. 119, pp. 86–101, 2017.

[13] "SLICES Test-Beds," https://portal.slices-sc.eu/, Accessed, October 2023.

[14] I. Baldin, A. Nikolich, J. Griffioen *et al.*, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Comput.*, vol. 23, no. 6, pp. 38–47, 2019.

[15] "PlanetLab Test-bed," https://planetlab.cs.princeton.edu/, Accessed, October 2023.

[16] "EdgeNet project," https://www.edge-net.org/, Accessed, October 2023.

[17] L. Mamatas, S. Skaperas, and I. Sakellariou, "ClusterSlice: Slicing Resources for Zero-touch Kubernetes-based Experimentation," University of Macedonia, Technical Report, November 2023.

[18] L. F. Gonzalez, I. Vidal, F. Valera *et al.*, "Link layer connectivity as a service for ad-hoc microservice platforms," *IEEE Netw.*, vol. 36, no. 1, pp. 10–17, 2022.

[19] G. Koukis, S. Skaperas, I. A. Kapetanidou *et al.*, "Performance evaluation of kubernetes networking approaches across constraint edge environments," 2024.

[20] B. C. Şenel, M. Mouchet, J. Cappos *et al.*, "Edgenet: a multi-tenant and multi-provider edge cloud," in *Proc. 4th ACM Int. Workshop Edge Syst., Analytics and Netw. (EdgeSys)*, Edinburgh, Scotland, UK, 26 Apr. 2021, pp. 49–54.

[21] B. C. Şenel, M. Mouchet, J. Cappos *et al.*, "Demo: Edgenet, a production internet-scale container-based distributed system testbed," in *2022 IEEE 42nd Int. Conf. on Distrib. Comput. Sys. (ICDCS)*, 2022, pp. 1298–1301.

[22] S. Skaperas, L. Mamatas, and A. Chorti, "Real-time video content popularity detection based on mean change point analysis," *IEEE Access*, vol. 7, pp. 142 246–142 260, 2019.

[23] A. Aue, C. Dienes, S. Fremdt *et al.*, "Reaction times of monitoring schemes for ARMA time series," *Bernoulli*, vol. 21, no. 2, pp. 1238–1259, 2015.

[24] R. Alami, O. Maillard, and R. Féraud, "Restarted Bayesian online change-point detector achieves optimal detection delay," in *Int. Conf. Mach. Learn.* PMLR, 2020, pp. 211–221.

[25] A. Aue and L. Horváth, "Structural breaks in time series," *J. Time Series Anal.*, vol. 34, no. 1, pp. 1–16, 2013.

[26] S. Skaperas, G. Koukis, I. A. Kapetanidou *et al.*, "A pragmatical approach to anomaly detection evaluation in edge cloud systems," *arXiv preprint*, 2024.