IoMiRCA: Root cause analysis in IoT-extended 5G microservice environments

Zeno Heeb, Onur Kalinagac, Wissem Soussi and Gürkan Gür
Institute of Applied Information Technology (InIT)
Zurich University of Applied Sciences (ZHAW)
Winterthur 8401, Switzerland
heebzen1@students.zhaw.ch,{name.surname}@zhaw.ch

ABSTRACT

Softwarized services in converged networks are evolving from monolithic applications to distributed architectures, often comprising numerous microservices. At the same time, with the massive proliferation of IoT devices, much more complexity and diversity are added to such critical infrastructures. In that regard, Root Cause Analysis (RCA) is an important part of a running distributed service ecosystem to keep the applications available and manageable by finding the root causes of errors and malfunctions. This paper provides a topology graph based anomaly detection and RCA solution for the microservice architecture in edge-to-cloud environments entailing microservices in combination with IoT.

CCS CONCEPTS

• **Networks** → **Network manageability**; **Mobile networks**; *Network performance analysis*; *Network reliability*.

KEYWORDS

Root cause analysis, critical infrastructure management, microservices, 5G and Beyond, edge-to-cloud continuum

ACM Reference Format:
Zeno Heeb, Onur Kalinagac, Wissem Soussi and Gürkan Gür. 2023. IoMiRCA: Root cause analysis in IoT-extended 5G microservice environments. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-April 2, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3555776.3577840

1 INTRODUCTION

Supported by the evolving ICT infrastructure and novel network technologies, the deployment of connected services is moving from physical machines to virtual ones (VMs) as well as containers. Additionally, modern services are in general split into multiple smaller services that communicate with each other for better scalability and granular management. In future networks like 6G, this architectural pattern is expected to become dominant with the emergence of novel concepts such as cloud-native operation, highly-specialized networks, edge-cloud continuum, and Native AI.

Concurrently, communication and networking infrastructure has emerged as a key critical infrastructure for all human activities ranging from digitally connected services such as e-banking to leisure activities such as streaming. This phenomenon was amplified with the recent COVID-19 lock-downs, which led to ubiquitous remote working and consumption of connected services over wired and wireless networks [2]. To identify a faulty service in a such a diverse, pervasive and fragmented system for service and network management can be time-consuming and thus calls for automated approaches [3]. This work proposes a Root Cause Analysis (RCA) scheme, namely *IoMiRCA*, running on top of a 5G infrastructure in a Kubernetes environment. We focus on a service architecture that includes Internet of Things (IoT) devices closely integrated into the network as typical in 5G and future network use cases [4] and leverage the extended MUD (Manufacturer Usage Description) file provided by a recent proposal called TRAILS [1]. The MUD standard essentially contains information about the allowed connection between domains and ports or further restrictions of the communication protocols [5]. Since the graph-based RCA approaches are widely applicable and provide good results, we focus on a graph-based approach in this work. To develop our IoMiRCA system, we use the MicroRCA [7] as a base approach and then broadly extend it for our scheme.

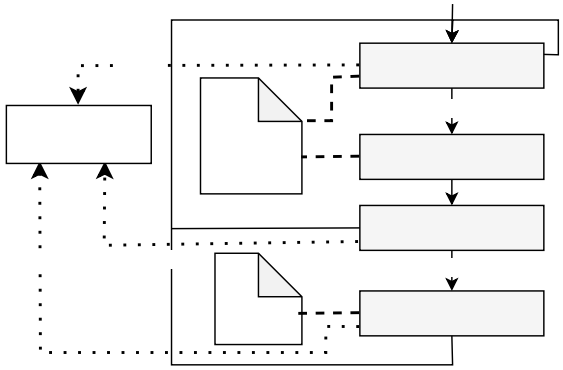

Figure 1: IoMiRCA algorithm and used data.

2 SYSTEM IMPLEMENTATION

2.1 IoMiRCA architecture

The procedure shown in Figure 1 starts with the collection of clean base data, necessary for the RCA algorithm to execute. The base data thereby only have to be collected once at the beginning before the rest of the algorithm loops as long as the IoMiRCA run. They

SAC '23, March 27-April 2, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03.
https://doi.org/10.1145/3555776.3577840

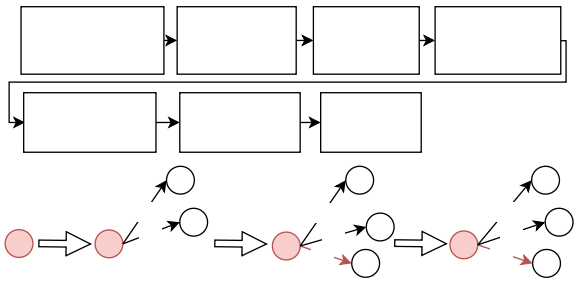

Figure 2: Detailed procedure of the RCA algorithm.

are later updated with new values in an improved sliding window approach. In the next step, we have the event handling part, which is for automatic fault injection into the services and therefore part of the test environment. It does not have a direct impact on the rest of the algorithm. The next step is anomaly detection which uses BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) clustering algorithm on the current latency values in combination with the ones from the base data. Only if an anomaly is detected, the RCA is done. Otherwise, we loop back to the metrics collection and the next iteration starts. If we have an anomaly, the RCA is done using all the metrics from the base data as well as the extended MUD information to pinpoint the root cause. This MUD information is taken from the TRAILS [1] proposal that contains the extended MUD file from [6]. A more detailed description of this RCA approach can be found in Section 2.3.

2.2 Metrics collection

The metrics used in this approach are collected using the `Istio`[1] service mesh for information about the latency of the communication between microservices. Additionally, we include the CPU, memory and network usage on the container level, collected by Google's `cAdvisor`[2] and on the node level, collected by the `node-exporter`[3]. All these metrics are collected and stored by Prometheus[4], and later on requested by our algorithm.

2.3 IoMiRCA RCA algorithm

In the first step in Figure 2 the destination nodes of anomalous edges, detected in the anomaly detection part, are taken into the graph. In the second step, all the outgoing communication, as well as the connection to the host node of the K8s cluster, are also taken into the anomalous subgraph. In the third step, we calculate the edge weights. The weights for connections to host nodes are calculated considering the node metrics, while the other weights are based on the correlation to the anomalous edge. In the fourth step, we consider the IoT devices. Every connection from an anomalous service to or from an IoT device is taken into the subgraph (including the IoT device itself). In the fifth step, all the other connections to or from this IoT device are included in the subgraph because these connections could be MUD violations and therefore indicate something is wrong with the IoT device. The detailed description of the weight calculation is provided in Section 2.3.2. In the sixth step, we calculate the personalization weight for the PageRank algorithm. Thereby only the anomalous service, as well as the directly connected IoT devices to such a service, have this special weight as these are considered the most likely root cause. Finally, in the last step, we run the PageRank algorithm on our weighted anomalous subgraph with additional personalization values.

2.3.1 Personalization calculations. The calculation for the personalization weights in the MicroRCA is to multiply the average edge weight (in and out) with the correlation [7]. With their average edge weight approach, we observe the problem that if the anomaly comes from one communication exchange the value should not change independently of how many other good connections exist. Furthermore, we have a different subgraph since we ignore the outgoing edges. To find a better solution, we tested four different methods with advantages in specific scenarios.

(1) Use the BIRCH algorithm which was also used for anomaly detection to detect some anomalies in the container and node level metrics. If there is an anomaly the personalization value is set higher since this consolidates the suspicion that something is wrong with this service. This is used in combination with the max value of the outgoing edges.
(2) Use the correlation between the latency of the communication and the container/node level metrics.
(3) Use the correlation from the method before but in a combination with the maximum value of the outgoing edge to include the propagation of the anomaly.
(4) Use a simple check condition to detect if there is a big change in the metrics compared to the metrics from the last loop. If such a change is detected, this results again in a higher personalization value since this is not the normal behaviour of the service. This penalty is then again combined with the max value of the outgoing edges.

2.3.2 Personalization and edge weight calculation for IoT devices. For non-IoT services, the weights of the edges in the graph are calculated based on the information if it is a connection to a host node (node metrics considered), if it is a communication that is discovered as anomalous in the anomaly detection part (α, a parameter that is used for setting weights and personalizations to an anomalous value by using this theoretical threshold, is taken as weight) or if it is a not anomalous connection to a service (correlation of latencies). Thereby α is the weight of an anomalous edge that has to be fine-tuned for each environment (default: 0.55).

Because of the missing communication and device resource metrics mentioned in Section 2, we have to use a different approach to define the weights of communications to an IoT device. Instead of using the communication information as well as the device resources we focus on the information given by the MUD and the connected services:

(1) If the communication is not supposed to be there according to the MUD rules, the weight is increased since this indicates that something is going wrong on the device.
(2) If a service is anomalous and very likely the root cause according to the data, this situation reduces the probability

[1] https://istio.io/
[2] https://github.com/google/cadvisor
[3] https://github.com/prometheus/node_exporter
[4] https://prometheus.io/

of the IoT device being the root cause. This works also conversely where the probability of the IoT is higher if there is no indication that the service is the root cause. Furthermore, if multiple services are anomalous to an IoT device this also increases the probability that something is going wrong with the IoT device.

The same reason for the missing information also counts for the personalization calculation. As for the weight calculation, we do not have any information for the default personalization calculation and then have to rely on the information we have for the connected services. Therefore the personalization of an IoT device is calculated considering the personalization value of all connected services. This generally results in a personalization value if at least one connected service is anomalous. But this is intended since IoT devices are considered as critical in our environment. Additionally, the MUD rules are also considered to add some penalty value if a violation occurs.

3 PERFORMANCE EVALUATION

3.1 Testbed

In our testbed, different OpenStack VMs are deployed as virtual network functions (VNF) using the Open Source Mano (OSM). In this cluster, we further deploy an adapted version of the sock-shop environment that consists of multiple microservices that communicate with each other. The most important adaption to the sock-shop environment is the additional services to include and simulate an IoT device. This simulated device sends the temperature to the microservices and as a result, changes the prices in the shop. User traffic and fault generation (response latency, memory and CPU stress) on containers are performed using `locust`[5] tool and the `Chaos Mesh`[6] platform, respectively.

3.2 Experimental Results

In Table 1, you can see an extract of calculated personalization values and suggested root causes from our IoMiRCA while doing fault injections on three services. In Table 1, 'C' is used for the `catalogue` service, 'T' for the IoT device (`temperature-sensor`) and 'I' for the `iot-handler` service. Thereby we have done each RCA instance once with the `evil-iot-handler` deactivated (no MUD violation) and once enabled (MUD violation). Furthermore, four different personalization functions (Pn) mentioned in Section 2.3.1 are used in these calculations but for the sake of brevity, only the results for the fourth approach are shown in Table 1.

As shown in Table 1, the algorithm works with the fourth approach pretty accurately to analyse the root cause if there is no IoT device in the anomalous subgraph. This is also the case for other approaches. However, as soon as we have IoT device(s) in the anomalous subgraph, the accuracy of the root cause prognosis gets lower, and more often, the actual root cause does not have the highest value. While the first and fourth personalization approaches are similar in their base structure, they differ in detecting the hardware "correlation" to the anomaly. The first approach with the BIRCH algorithm is slower and less stable. This instability occurs in the form of detected anomalies in the container metrics that are not strongly related to the analysed problem. The second and third approaches are both partially based on the correlation, and therefore, we get similar results. Nevertheless, both of these approaches have less range in the results and the IoT device is generally placed further down the suggested root cause list. The most complicated scenario considering all tests is the situation when the `iot-handler` is the root cause, and the IoT device has communications that violate the MUD rules.

[5]https://github.com/locustio/locust
[6]https://chaos-mesh.org/

Table 1: IoMiRCA algorithm results with four different personalization functions (Pn) in different fault scenarios.

Pn	Fault service	Evil	Personalization	Anomalies
4	Catalogue		C:0.63	C:1.0
		X	C:0.66	C:1.0
	Iot-handler		I:0.73 T:0.28	I:0.78 T:0.22
		X	I:0.22 T:1.33	I:0.29 T:0.50
	temperature- sensor		I:0.22 T:0.775	I:0.53 T:0.46
		X	I:0.22 T:1.33	I:0.29 T:0.50

4 CONCLUSION

This paper proposes an RCA scheme with integrated anomaly detection for a service environment in containerized networked applications. The experiments show that the algorithm successfully detects root causes in our environment where microservices exist alongside IoT devices. Due to missing IoT device metrics and the assumption that they are an unmanaged domain, our approach inherently provides limited suggestions that can be improved by extending the metric collection architecture as future work.

REFERENCES

[1] Yacine Anser, Chrystel Gaber, Jean-Philippe Wary, Sara Nieves Matheu García, and Samia Bouzefrane. 2022. TRAILS: Extending TOSCA NFV profiles for liability management in the Cloud-to-IoT continuum. In *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. 321–329. https://doi.org/10.1109/NetSoft54395.2022.9844027
[2] Broadband Internet Technical Advisory Group (BITAG). 2021. 2020 Pandemic Network Performance. http://www.bitag.org/documents/bitag_report.pdf.
[3] Chrystel Gaber, José Sánchez Vilchez, Gürkan Gür, Morgan Chopin, Nancy Perrot, Jean-Luc Grimault, and Jean-Philippe Wary. 2020. Liability-Aware Security Management for 5G. In *2020 IEEE 3rd 5G World Forum (5GWF)*. 133–138. https://doi.org/10.1109/5GWF49715.2020.9221407
[4] Zeno Heeb, Onur Kalinagac, Wissem Soussi, and Gürkan Gür. 2022. The Impact of Manufacturer Usage Description (MUD) on IoT Security. In *2022 1st International Conference on 6G Networking (6GNet)*. 1–4. https://doi.org/10.1109/6GNet54646.2022.9830354
[5] E. Lear, R. Droms, and D. Romascanu. 2019. *Manufacturer Usage Description Specification*. RFC 8520. RFC Editor. https://doi.org/10.17487/RFC8520
[6] Sara Nieves Matheu, José Luis Hernández-Ramos, Salvador Pérez, and Antonio F. Skarmeta. 2019. Extending MUD Profiles Through an Automated IoT Security Testing Methodology. *IEEE Access* 7 (2019), 149444–149463. https://doi.org/10.1109/ACCESS.2019.2947157
[7] Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. 2020. MicroRCA: Root Cause Localization of Performance Issues in Microservices. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9. https://doi.org/10.1109/NOMS47738.2020.9110353