Dynamic Service Placement in 6G Multi-Cloud Scenarios

Fatemeh Tabatabaei, Hamzeh Khalili, Manuel Requena, Sarang Kahvazadeh, and Josep Mangues-Bafalluy
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Castelldefels, Spain
e-mail: ftabatabaeimehr@cttc.es

ABSTRACT
The rapid development of 6G technology promises to revolutionize wireless communication and bring significant advancements in various industries. Public Protection and Disaster Relief (PPDR) applications require cutting-edge communication technology to meet their low latency, high-speed, and bandwidth requirements for effective decision-making support during emergencies. This paper presents a federated framework for the 5G-EPICENTRE project that allows various testbeds to federate and share resources for emergency situations. The proposed federated framework can provide flexible and efficient utilization of resources, enabling more effective PPDR applications to be connected and utilized. To further enhance the performance of PPDR applications, we propose a scheduler that optimizes Network Service (NS) placement for Cloud and Multi-access Edge Computing (MEC) resources to improve the overall performance of PPDR applications. Our approach aims to reduce latency and efficiently offload services to empower first responders to make critical decisions during emergencies. This approach is expected to improve the overall performance of the PPDR applications and empower first responders to make critical decisions in emergency situations.
Keywords: 6G, autonomous network management, multi-cloud, federation, Karmada, scheduler.

1. INTRODUCTION

In recent years, the growth in data traffic has created a need for faster and more reliable wireless communication technologies. While 5G technology is still being deployed worldwide, researchers and industry experts are already looking towards the future of wireless communication with 6G technology [1]. 6G promises to offer unprecedented data transfer rates, ultra-low latency, and enhanced connectivity, enabling new applications and services that are not possible with current wireless communication standards [2].

One sector that can greatly benefit from 6G technology is the public protection and disaster relief (PPDR) sector. PPDR agencies need high-speed, reliable communication networks to effectively respond to emergencies and natural disasters. However, existing communication networks [3] can often be disrupted or overloaded during critical situations, hindering the ability of PPDR agencies to respond quickly and efficiently. With the implementation of 6G technology, PPDR agencies can benefit from advanced technologies such as real-time situational awareness, augmented reality, and autonomous systems, which can enable them to respond more efficiently and effectively. Deploying multi-domain network services in a 6G scenario requires advanced solutions, such as service federation [6]. Federation is the process of orchestrating services or resources across several domains in a multi-domain scenario.

The Horizon 2020 5G-EPICENTRE project [5] defines an architecture of the experimentation facilities provided by different partners of the project, with support for cross-domain and cross-testbed experiments. In addition, new technological evolution is required to pave the way for endless services tailored to specific verticals like use cases for PPDR [7]. In this direction, AI/ML approach like optimization techniques can be helpful for service placement and for efficiently offloading and directing traffic between the Cloud and Multi-access Edge Computing (MEC) resources available [7]. Therefore, in this paper, we describe a cross-testbed federation environment and a new scheduler designed in federation to consider the resources available at the network edge and across multiple geographically distributed testbed infrastructures, providing interconnectivity among them, which is crucial for effective communication and coordination during emergency situations. The paper is organized as follows: Section 2 overviews the Karmada architecture. Section 3 describes cross-testbed federation approach, Section 4 presents the novel scheduler framework. Section 5 concludes the paper.

2. BACKGROUND

Karmada [8] is an open-source, multi-cloud platform that enables Kubernetes orchestration by utilizing Kubernetes-native APIs and advanced scheduling capabilities. It offers automation for managing multi-cluster applications in both hybrid and multi-cloud scenarios, with features such as centralized multi-cloud management, high availability, failure recovery, and traffic scheduling. Karmada's architecture is similar to that of a single Kubernetes cluster, with a control plane, API server, scheduler, and controllers. The Karmada Control Plane is presented in Fig. 1 as part of the 5G-EPICENTRE architecture, and it consists of multiple components, including the API server, Karmada cluster controllers, Karmada scheduler, and synchronization modes. Karmada features the following fundamental components:

- The API server of the control plane provides Kubernetes-native APIs, policy APIs, and persist metadata in ETCD.
- Karmada cluster controllers are the core component of the Karmada system. They are responsible for managing the lifecycle of a deployed service from submission to final deployment, maintaining the federated cluster, and managing objects for different purposes, such as maintaining the cluster status, workload, and policies.
- The Karmada scheduler [9] provides the interface that defines the fundamental functionalities of scheduling a deployment. Currently, it supports cluster affinity, cluster locality, API enablement, taint toleration, and spread constraints.
- Karmada supports two modes of coordination (push and pull) for a given member cluster. It maintains the member cluster directly so that all objects and namespaces are created inside the Karmada control plane. Alternatively, Karmada can create an agent server in the member cluster's workspace as the delegate of the Karmada API server [8].

3. CROSS-TESTBED FEDERATION ARCHITECTURE

The idea of cross-testbed federation [10] is conceptualized in Fig. 1, and it enables multi-clustered Kubernetes (K8s) orchestration across different domains, allowing testbed federation [10]. K8s-based orchestrators manage and provide access to service resources for individual testbed containerized workloads, facilitating multi-clustered K8s orchestration across different domains.

To achieve this, Karmada is used, enabling cloud-native applications to run across multiple K8s clusters without any changes to the application. Karmada provides a centralized control plane for application deployment and resource management on multiple clusters, known as member clusters. Karmada also allows the federation of any K8s resources in a multi-cluster environment. Initially, Karmada provides a cluster federation across different testbed infrastructures by creating a new abstraction of a federation layer. However, additional functionalities can be developed for end-to-end service across 5G-EPICENTRE testbed infrastructures to meet the architecture requirements of 5G-EPICENTRE.

To integrate the individual testbeds into the 5G-EPICENTRE federation and facilitate system communication mechanisms, the Message Queuing Telemetry Transport (MQTT) protocol is deployed for asynchronous communication using RabbitMQ. The components of 5G-EPICENTRE can communicate in both synchronous and asynchronous modes [11].

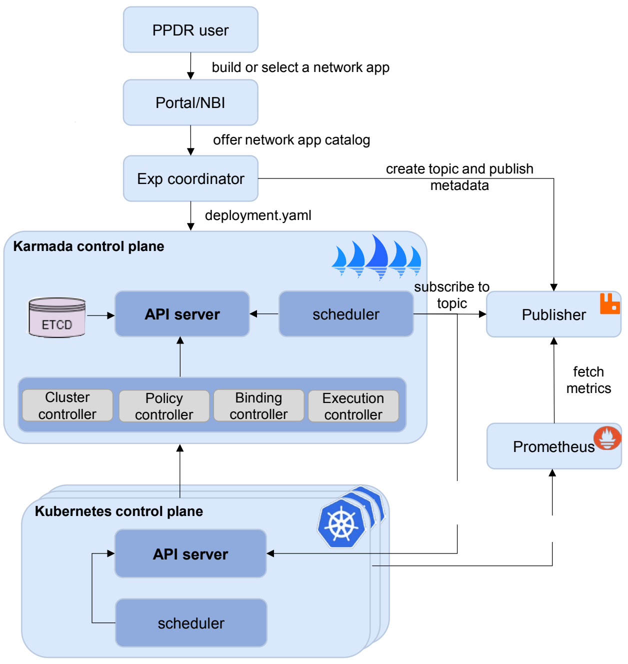

Figure 1. 5G-EPICENTRE architecture.

At a high level, the workflow for managing the service involves several key modules, with Karmada serving as the cross-testbed container orchestrator that enables seamless management of running cloud-native applications across different infrastructures. In particular, the first three modules in Fig. 1 are focused on developing a solution that enables the connection between PPDR service providers and the 5G-EPICENTRE platform, as well as between the portal and cross-testbed federation. Once a service is submitted, the binding controller interacts with the Karmada API server to create the initial binding object based on the policies defined by the user. These policies indicate the strategies considered for propagating the pods across clusters. The Karmada scheduler focuses on fault-domain and includes a nested function to check the validity and reliability of member clusters. It has an in-tree plugin structure that separates the core process from the sub-components. In general, a pluggable scheduler framework allows defining sub-functions of different strategies and controls policies that affect the scheduling behavior. As we outlined in the introduction, the scheduler should be able to fulfill the required KPIs from the use cases. To achieve this aim, the scheduler fetches the required metrics by connecting to a publisher that is configured to use the Message Queuing Telemetry Transport (MQTT) protocol or by using a client API that exposes the Kubernetes cluster API server. Moreover, the publisher integrates with Prometheus, which allows monitoring service metrics from the infrastructure. Once the scheduler indicates the target cluster, the execution controller places the workload in the corresponding cluster namespace. After this step, the Kubernetes API server is triggered to schedule the pod on the most appropriate node. Note that for this project, we have also made contributions to the Kubernetes scheduler.

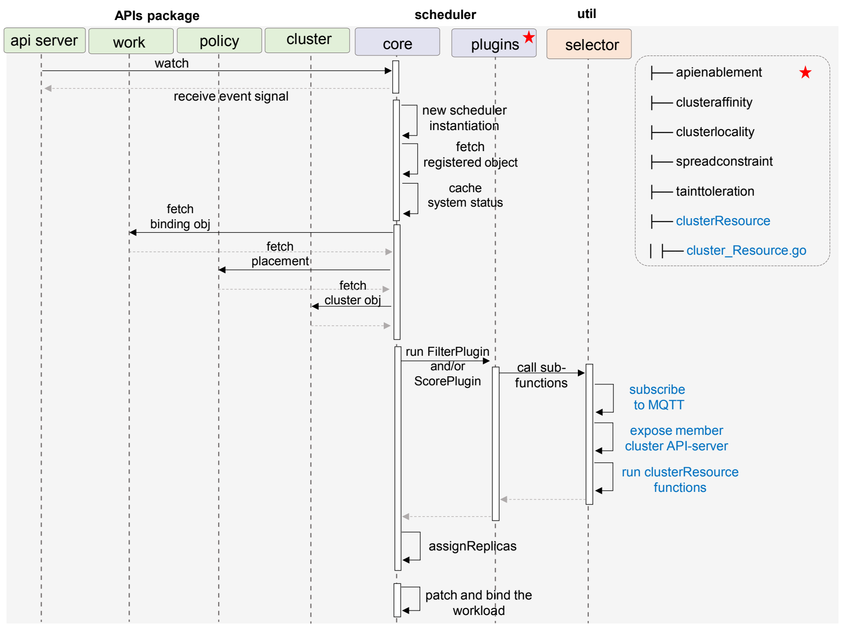

Figure 2. Service lifecycle in Karmada's scheduler.

4. CUSTOMIZED SCHEDULING FRAMEWORK

To achieve the goals outlined in the introduction, we designed and experimentally developed a new plugin, called *clusterResource*, which is attached to the standard scheduler in the Karmada system. The plugin guarantees the fulfillment of required KPIs by computing the optimal placement for a given service. Although the new plugin is decoupled from the service and infrastructure, it accesses the necessary metrics required for scheduling the PPDR service, mainly latency. The *clusterResource* plugin, like other native plugins in the Karmada scheduler, executes some key tasks, which are depicted in blue in Fig. 2, and the core contributions are implemented in the plugin and selector modules.

The scheduler uses a watch API as a client to the Karmada API server and is notified about changes in the resources (see Fig. 2). Once the scheduler is triggered, it creates a new instance of the scheduler struct and takes two parameters: i) cache and ii) registry, which are the instances of Cache interface and runtime Registry interface, respectively. The scheduler then calls Filter Plugin and/or Score Plugin, depending on the defined policies or which ones are enabled. *clusterResource* only uses FilterPlugin interface. To provide context before diving into the details, we will first explain the process of adding a new plugin to Karmada's scheduler. The *clusterResourc*e template is similar to the native plugins template as is marked with star in Fig. 2. Once the template is created, the FilterPlugin is added to fix the arguments required for scheduling process. Afterward, the new template should be registered in the list of available plugin inside the main implementation of Karmada scheduler so that it will then become a part of the scheduling framework, making it available once a workload is created.

The *clusterResource* plugin uses a client API to expose the K8s' API server to query the allocatable resource of each member cluster. On the other side, it subscribes to a specific topic published in MQTT to fetch service metric. Let's assume that the available resource of a cluster $c \in C$ is a summary of node metric, and E2E latency is aggregated internal latency along the service chaining plus the external latency from the ingress node to the end user (EU). The *clusterResource* algorithm employs a linear programming (LP) approach to solve the placement problem, which can minimize the average latency in the long term while not exceeding the metrics limitation of constraints. Since the service placement is categorized as a nondeterministic polynomial (NP)-hard problem, when the size of input is scaling up, the complexity of the problem also increases exponentially. To avoid this complexity, we proposed a combination of a local search algorithm and an analysis function to consider the long-term behavior of a cluster. In this way, the cluster c_{target} with the best condition to host the service is selected.

Afterward, the assignReplica function specifies the target clusters for a given object based on the replica scheduling strategy. Note that if no cluster is found at this step, the event triggers the de-scheduling module, which is a separate package in the Karmada project. Finally, the Binding method calls sub-methods based on the type of placement policy to patch the placement and the list of ResourceBinding objects after it has been scheduled.

5. CONCLUSIONS

Karmada allows the federation of multiple K8s clusters across different testbed infrastructure of the project and allows the centralized Karmada controller to coordinate multiple clusters and applications that can be deployed in multi-cluster environments and different domains. In addition, we proposed a solution to solve the service placement in EC-enabled 6G networks, considering the latency and computational resources that makes it suitable to solve the stringent requirements of PPDR use cases but can be of general application.

ACKNOWLEDGEMENTS

This work has been funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101016521 5G-EPICENTRE.

We would like to acknowledge Hongcai Ren and Karmada community for adding the capability to extend the built-in plugin in the scheduler. Their willingness to help us with our specific needs was instrumental in the successful completion of this work.

REFERENCES

[1] Zhang, Y., Jiang, T., Feng, G., Zhang, Y., Wang, W., and You, X.: An overview of 6G research progress and prospect, *IEEE Transactions on Vehicular Technology*, 70(3), 2908-2926, 2021.
[2] T. Taleb, Z. Pang, J. Li, *et al.*: 6G wireless networks: Vision, requirements and challenges, *IEEE Network*, vol. 34, no. 3, pp. 487-493, May/Jun. 2020.
[3] F. Boccardi, R. W. Heath Jr, A. Lozano, *et al.*, "Five disruptive technology directions for 5G, *IEEE Communications Magazine*, vol. 52, no. 2, pp. 74-80, Feb. 2014.
[4] A. S. K. Pathan, et al.; Multi-domain federation of 5G and cloud services for PPDR, in *Proc. 2020 IEEE 3rd 5G World Forum (5GWF)*, pp. 219-224, Sep. 2020.
[5] A. Dimitrios *et a*l. "Unification architecture of cross-site 5G testbed resources for PPDR verticals, in *Proc. 2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom).* IEEE, 2021.
[6] J. Baranda Hortiguela *et al.*: Realizing the network service federation vision: Enabling automated multidomain orchestration of network services, *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 48-57, Jun. 2020.
[7] C. R. de Mendoza, B. Bakhshi, E. Zeydan, and J. Mangues-Bafalluy: Near optimal VNF placement in edge-enabled 6G networks, in *Proc. 2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, Paris, France, 2022, pp. 136-140.
[8] Karmada: Open, Multi-Cloud, Multi-Cluster Kubernetes Orchestration. available at: https://karmada.io/
[9] Karmada Scheduler. Retrieved April 19, 2023. Available at: https://karmada.io/docs/karmada-scheduler/.
[10] 5G-EPICENTRE, "D4.4 5G-EPICENTRE Experimentation facility preliminary version", Jun. 2022.
[11] 5G-EPICENTRE, "D1.1 5G-EPICENTRE experimentation scenarios preliminary version", Jun. 2021.