Edge-Aware DNS Routing in Kubernetes for Multi-Region Deployments

Anila Gogineni

Independent Researcher
USA
anila.ssn@gmail.com

Abstract
Organization success with edge computing and multi-region Kubernetes depends on proper traffic optimization because it makes services faster and stays online more often. The standard DNS lookup approach creates poor network routes because it lacks real-time information about network speed and uses up resources needlessly. This survey explores the role of edge-aware DNS routing in Kubernetes for multi-region deployments, emphasizing its impact on latency reduction, load balancing, fault tolerance, and scalability. It examines key challenges, including infrastructure complexity, security vulnerabilities, and data consistency, while highlighting solutions such as topology-aware routing, real-time network measurements, and orchestration strategies. The study reviews Kubernetes' networking components, including Core DNS, for service discovery and its integration with edge computing to optimize performance in 4G and 5G environments. By analyzing existing approaches, this survey provides insights into best practices and future research directions for enhancing DNS-based traffic management in distributed cloud-edge architectures.

Keywords: Edge Computing, Kubernetes Orchestration, Multi-Region Deployments, DNS Routing, Service Discovery

I. INTRODUCTION

With the rapid growth of cloud-native applications, enterprises are increasingly deploying workloads across multiple regions to ensure high availability, low latency, and resilience[1]. Kubernetes, as the leading container orchestration platform, provides a robust framework for managing such deployments. However, ensuring efficient and reliable Domain Name System (DNS) routing across multiple regions is a challenge, particularly in edge computing scenarios where latency-sensitive applications require optimized traffic routing[2]. The traffic routing has become a bit complex with the rising adoption of edge computing especially when there are multiple regions of Kubernetes. Previous approaches of cloud architectures are to have centralized DNS resolution; this can result in added latency and poor response time in large and distributed user base environments. Edge-aware DNS routing resolves these problems with its method of sending traffic to the closest accessible region to deliver improved performance together with lower response times and better user experiences[3]. The open-source and free Kubernetes platform automates and manages the deployment, scaling, and operations of applications that are containerized. Among the most popular container management systems[4], Kubernetes has simplified container deployment by eliminating tedious manual tasks [5]. Kubernetes automates software distribution and container management for many IT organizations, including IBM, Pinterest, and the US Department of Defense (DoD) [6].

In edge computing, workloads are distributed closer to the end users to minimize latency and improve responsiveness. Traditional DNS routing approaches may not always route requests efficiently, leading to increased round-trip times and potential service degradation[7]. Edge-aware DNS routing dynamically selects the nearest edge location based on real-time network metrics, reducing response times and enhancing the user experience[8]. Deploying Kubernetes across multiple regions involves managing multiple clusters that can operate independently or as a federated system. Challenges such as cross-cluster service discovery, data consistency, and failover mechanisms require robust DNS-based solutions[9]. Global load balancers, geo-aware DNS services, and Kubernetes-native tools like External DNS help direct traffic effectively while maintaining high availability[10].

In multi-region Kubernetes deployments, ensuring efficient and reliable traffic routing is critical for optimizing performance, reducing latency, and improving user experience. Traditional DNS-based routing mechanisms often fail to account for dynamic network conditions, leading to inefficient traffic distribution and potential bottlenecks. Edge-aware DNS routing addresses these challenges by intelligently directing requests to the nearest or most optimal cluster based on real-time metrics such as latency, load, and availability. This approach enhances resilience, minimizes cross-region data transfer costs, and ensures high availability, making it a crucial strategy for enterprises operating distributed applications at a global scale. The main focuses of the study are as follows:

- Provides an in-depth review of edge-aware DNS routing strategies in Kubernetes, highlighting their role in improving service availability, latency reduction, and efficient resource utilization in multi-region deployments.
- Examines how edge computing principles enhance Kubernetes-based architectures by optimizing data processing, fault tolerance, and regulatory compliance in distributed environments.
- Discusses the role of Core DNS in Kubernetes networking, detailing how it facilitates efficient DNS resolution, service discovery, and load distribution across multiple clusters.
- Identifies key challenges, including infrastructure complexity, security risks, and data consistency issues, offering insights into strategies for mitigating these concerns.

Structure of the paper

This paper explores Edge-Aware DNS routing in Kubernetes for multi-region deployments. Section II provides an overview of Kubernetes networking; Section III discusses edge computing and its role in multi-region deployments; Section IV introduces edge computing in Kubernetes; Section V provides the Kubernetes and docker swarm container schedulers for edge computing; Sections VI and VII provide the literature review and conclusion with future scope.

II. OVERVIEW OF KUBERNETES NETWORKING

Microservices have streamlined cloud-native application development but introduce security challenges, especially in ensuring secure service interactions within highly distributed environments[11]. Even small cloud applications have multiple microservices, while larger systems may have hundreds or thousands, each running in separate containers. Kubernetes, a leading container orchestration platform, simplifies service management but abstracts security risks, making them easier to overlook. While cloud infrastructure and software-defined networking solutions have been examined for security, Kubernetes networking components remain underexplored. Key concerns include enforcing network policies, securing service connections, and mitigating vulnerabilities in networking plug-ins. Kubernetes' default settings prioritize connectivity over strict security controls, increasing the risk of misconfigured networks. Traditional security models rely on physical barriers like firewalls, whereas Kubernetes operates in a software-defined environment, requiring a new approach to network security[12]. The misconception of "digitally unbridgeable moats" in cloud networks can lead to overconfidence. Kubernetes itself is not inherently insecure, but applying outdated security concepts without adaptation poses significant risks. A modern, Kubernetes-specific security strategy is essential.

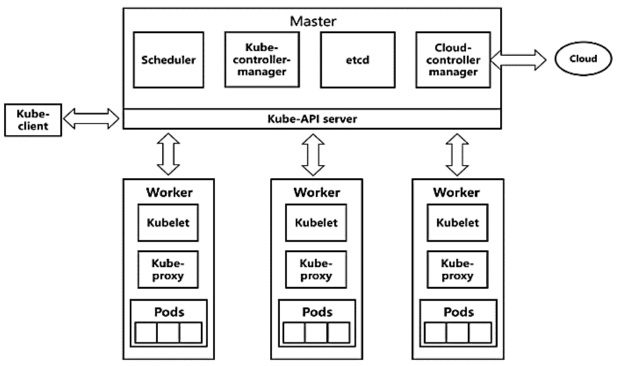


Fig. 1. Kubernetes Architecture

Kubernetes Architecture

Kubernetes' distributed design is based on the command-line tools offered by client, worker, and master nodes. Figure 1 shows the specific architecture of the Kubernetes diagram.

Master Node:

Every Kubernetes cluster revolves around the master node, which is in charge of application administration, scheduling resources, and distributing jobs. Schedule, API Server, ETCD, and Controller Manager are some of the most crucial master nodes. Finally, here's a little illustration of them:

- The API Server bridges the gap between the various Kubernetes components, processes all cluster elements, and provides services like adding, deleting, editing, and querying resource objects.
- The Kubernetes cluster's administrator and central node is the Controller Manager. The Controller Manager is responsible for promptly identifying and addressing any system anomalies.
- Scheduler is the default resource scheduler for the Kubernetes cluster. In order to allocate waiting pods to nodes that are expected, schedulers employ anticipated rules. The node schedule is determined by the Scheduler using API Server after the Controller Manager has generated a Pod object. The binding information is then published to ETCD.
- ETCD is a key-value store that the Kubernetes cluster uses to keep running smoothly and to store many kinds of cluster data with high availability and durability.

Worker Node

A worker node in a Kubernetes cluster is in charge of making use of the resources available to it, doing the tasks that the master node has assigned to it, and receiving and finishing tasks that are under the master's control. The two primary components of a worker node are:

- Container lifecycle management, job execution per master node instructions, and communication with all cluster nodes are all responsibilities of the Kubelet service processes. Kubelet also ensures that the master control node is always up-to-date on the operational state and resource use of all labor nodes.
- The Kube-proxy service process is executed by each worker node and is responsible for relaying the service's request for access to the backend.

DNS in Kubernetes

Services specified in a Kubernetes cluster may be located and named with the help of a DNS server that is part of the platform. The cluster also hosts this domain name server as a replicated service. The number of

DNS servers visible in your cluster will be proportional to its size. The DNS service is run as a Kubernetes deployment, stated in Figures 2 and 3, which manages these replicas:

```
$ kubectl get deployments --namespace=kube-system core-dns
NAME        DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
core-dns    1          1          1             1            45d
```

Fig. 2. Checks the deployment status of CoreDNS.

The DNS server's load balancing may be handled by another Kubernetes service:

```
$ kubectl get services --namespace=kube-system core-dns
NAME        CLUSTER-IP    EXTERNAL-IP    PORT(S)           AGE
core-dns    10.96.0.10    <none>     53/UDP,53/TCP     45d
```

Fig. 3. Retrieves details about CoreDNS service.

The DNS service for the cluster is associated with the IP address 10.96.0.10. This has been added to the /etc/resolv.conf file of each container in the cluster. You can see this by logging into one of the containers.

III. EDGE COMPUTING AND ITS ROLE IN MULTI-REGION DEPLOYMENTS

Computing at the network's perimeter, on data that is either upstream for IoT services or downstream for cloud services, is made possible by a collection of technologies known as edge computing. In [13], ‘Edge computing is a concept that moves to process closer to the source of data, allowing for more sophisticated localization’. As defined in [14], ‘The next big thing in computing, "edge computing," is moving processing power closer to the gadgets that actually use it. Figure 4 depicts multi-access edge computing, a use case of edge computing that aims to improve user experience via consistent service delivery and highly efficient network operations.

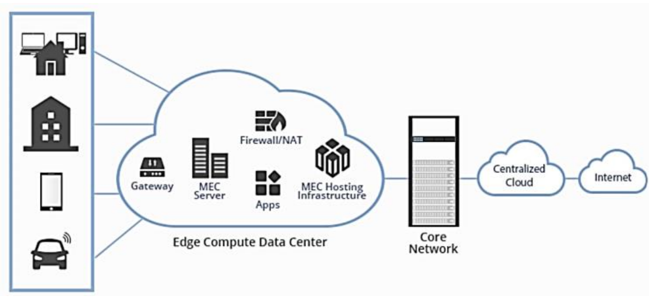


Fig. 4. Multi-access edge computing

It may not be a new idea to situate data centers near the sources of such data [15]. In 2002, the phrase "edge computing" was coined to describe the logical relocation of applications serving from cloud data centers to the periphery of a network, as seen from a business standpoint. The word was later employed in 2004 to describe a system that improved the system's performance by distributing program techniques and data to the network's edge [16]. When cloud computing was limited to distributing and hosting resources in central data centers, edge computing continued to advance and eventually solved many of cloud computing's issues by providing elastic resources to end users at the network's periphery[17].

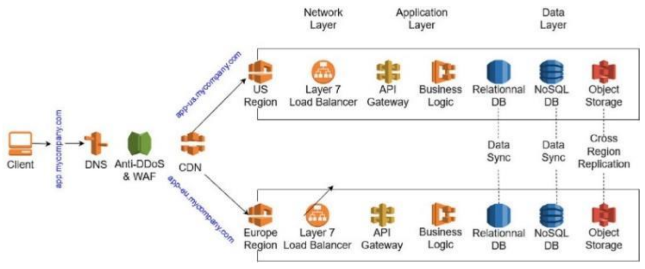


Fig. 5. Example of multi-region architecture

Figure 5 illustrates a multi-region architecture using AWS services designed for high availability and disaster recovery. A user accesses the application through a globally routed URL (e.g., app.mycompany.com), which utilizes DNS and a Content Delivery Network (CDN) for optimized routing and caching. Traffic is directed to a regional Application Load Balancer (Layer 7) in either the US or Europe, based on user location. Each region operates independently with its own stack, including API Gateway, business logic, relational and NoSQL databases, and object storage. Data synchronization and cross-region replication mechanisms ensure data consistency across regions. Security measures like Anti-DoS and WAF are implemented at the entry point (CDN) to protect the application. This architecture allows the application to remain operational even if one region experiences an outage, as traffic can be directed to the healthy region.

Benefits of Edge Computing in Multi-Region Deployment

Edge computing is essential in a multi-region Kubernetes implementation for optimizing resource utilization, increasing fault tolerance, and bettering the user experience. Some key benefits include:

- **Reduced Latency:** Edge computing reduces the latency of long-distance data transmission by bringing processing of data closer to end consumers.
- **Bandwidth Optimization:** Reducing data transmission requirements to cloud servers makes network bandwidth more efficient [18].
- **Improved Reliability:** Operation continues when an outage impacts central cloud connectivity.
- **Scalability:** The system allows automatic resource distribution and scaling adjustments through local usage requirements.
- **Compliance and Data Sovereignty:** The practice allows organizations to fulfill regulatory mandates by keeping sensitive information inside particular geographical zones.

Challenges of Edge Computing in Multi-Region Deployments

Despite its advantages, edge computing introduces several challenges:

- **Infrastructure Management:** Running edge nodes in multiple regions to manage is a complex task.
- **Security Risks:** The security risks faced by Edge devices increase because these devices connect to external networks [19].
- **Data Consistency:** The successful operation between edge nodes and main cloud locations faces difficulties when maintaining consistent data synchronization.
- **Resource Constraints:** Traditional cloud data centers possess superior computational and storage capabilities than the basic functions available in edge nodes.

Edge-Aware DNS Routing for Multi-Region Deployments

Edge-aware DNS routing enhances multi-region Kubernetes deployments by directing user requests to the nearest or most optimal edge location. This approach leverages:

- **Geo-aware DNS:** The system directs requests according to users' location by connecting to the nearest edge node.
- **Latency-based Routing:** Directs traffic to the edge region with the lowest response time[20].
- **Load-Aware Balancing:** The system performs intelligent traffic management through the evaluation of edge node operational limits and active readiness.
- **Failover Mechanisms:** Redirects traffic to alternate regions in case of failures, ensuring high availability.

Edge-Aware DNS Routing for Multi-Region Deployments

Edge-aware DNS routing enhances multi-region Kubernetes deployments by directing user requests to the nearest or most optimal edge location. This approach leverages:

- **Geo-aware DNS:** Routes requests based on the user's geographical location to the closest edge node.
- **Latency-based Routing:** Directs traffic to the edge region with the lowest response time[21].
- **Load-Aware Balancing:** Dynamically distributes traffic based on edge node capacity and availability.
- **Failover Mechanisms:** Redirects traffic to alternate regions in case of failures, ensuring high availability.

Benefits of Edge-Aware Architectures

This section provides a benefit of edge-aware architecture as follows:

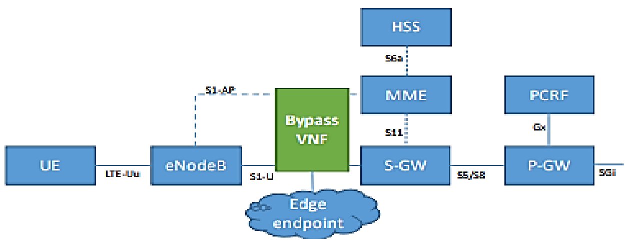
- **Reduced Latency:** Applications that are sensitive to latency have found great success using edge computing, which simplifies network architecture while simultaneously reducing data transmission times [22].
- **Mobility enhancement:** It is common practice to use a network of geographically dispersed fog and edge devices for processing and storage in an architecture characteristic of edge computing.
- **Ease of data processing:** Edge computing can analyze and extract valuable insights from "big data" due to its capabilities of being implemented near data sources.
- **Location Awareness:** The edge computing model differs from cloud computing in that it makes use of data's physical location to do certain computations.

IV. INTRODUCTION TO EDGE COMPUTING IN KUBERNETES

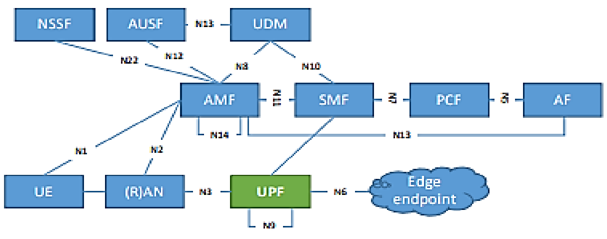
Computing at the network's edge offers powerful processing power, dependable connectivity, and real-time capabilities to end users' devices. Container technology primarily enables the delivery of resources and workloads to the edge. A persistent area of concern is the optimal timing, location, and method for delivering containerized workloads. To address these concerns, Kubernetes has emerged as the leading technology for containerized service orchestration. Despite Kubernetes's shortcomings, such as its inability to employ topology awareness and real-time network measurements for scheduling, it is nonetheless used to realize cloud-edge systems[23].

Edge computing deployment and issues

The importance of edge computing in meeting low latency requirements has been acknowledged by many [24]. This paradigm is therefore considered a cornerstone of 5G, but the original ETSI MEC standard design was really tailored to function with any mobile network. This heterogeneity enables all parties in the mobile ecosystem to profit from Edge Computing's advancements in the telecom industry while continuing to depend on 4G networks, as shown in Figure 6.



(a) 4G



(b) 5G

Fig. 6. Realization of the Edge Computing attach point (a) in 4G and (b) in 5G networks.

The inclusion of Edge Computing in 4G is anticipated to pave the way for applications requiring reduced latency and/or localization. Additionally, it is considered a "gateway" to the 5G network's improved infrastructure and applications. The need to build Edge Computing solutions as an add-on to existing 4G networks to provide edge services creates a variety of challenges, including user-plane integration and tenant space separation. Without the ability to link application components operating in various data centers to UEs and between themselves, edge computing technology cannot provide a significant decrease in latency times while maintaining firm service continuity standards. However, as previously noted by the ETSI MEC Working Group (WG), it is possible for NFV services and application user planes to be hosted in separate, segregated VIM tenant spaces. To facilitate the interchange of user-plane communication between two separate tenant spaces, so-called "attach points" must be present. VNFs and application components housed in the VIM are connected by virtual networks, which provide attachment points. This makes them critical for application-network service provider user-plane traffic exchange. Nevertheless, it is not easy to realize attach points across NFV services and applications due to the fact that different stakeholders operate in different tenant regions and may have different access/privacy requirements.

V. KUBERNETES AND DOCKER SWARM CONTAINER SCHEDULERS FOR EDGE COMPUTING

Cloud and edge computing both use Kubernetes as their primary container orchestration technology. Created by Google, it has a sizable open-source community that helps keep it running. Google Cloud, Amazon Web Services, RedHat, and Microsoft Azure are just a few of the cloud computing platforms that incorporate Kubernetes. Kubernetes streamlines the process of deploying, scaling, and administering applications that use containers. An important part of the Kubernetes framework is the Kubernetes scheduler. Containers, often called pods, are assigned to edge nodes by the scheduler. A framework for making decisions based on several factors is used, and it is greedy. The most basic kind of node labeling uses a weighted combination of fixed metrics like CPU and memory. The scheduler takes into account other node- and task-related parameters such as data location, deadlines, inter-workload interference, affinity and anti-affinity requirements, and hardware/software/policy limitations. Figure 7 visually represents a Docker Swarm service deployed across multiple hosts. The "Swarm Manager" residing on "Node - 0" manages the cluster and hosts the "Overlay Network," facilitating communication between containers across different hosts. A "Swarm service" is defined, which dictates the desired state and configuration of the application. This service is then implemented by running multiple containers ("Container-1," "Container-2," up to "Container-N") across different hosts ("Host-1," "Host-2," up to "Host-N"). This distribution of containers across hosts provides redundancy and scalability for the application. The diagram highlights how Docker Swarm orchestrates containers to fulfill the service definition, leveraging the overlay network for inter-container communication regardless of the host they reside on.

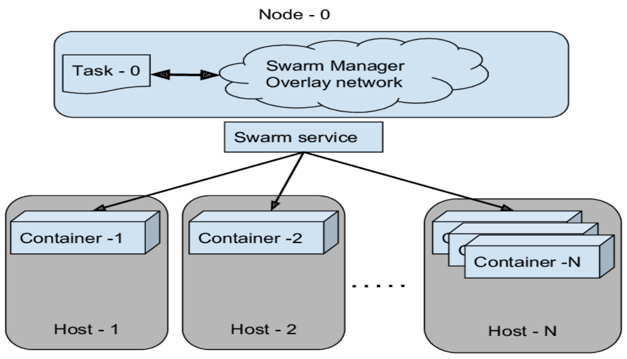


Fig. 7. Docker Swarm service containers spread across an overlay network

When determining the optimal location for containers, the Kubernetes scheduler takes various measurements and parameters into account. The scheduler in Kubernetes may be customized using unique metrics and variables. Pods may provide their metrics to the control plane, the central server in Kubernetes. This allows for the monitoring of resource utilization at each edge node at regular intervals, similar to heartbeats. Among other things, the scheduler uses this data to arrange pods in a manner that distributes workload uniformly and meets other criteria. Additionally, the taint/tolerance and predicates/priorities feature of the Kubernetes scheduler are used to limit the pods that may be assigned to an edge node. Outline the features of the Kubernetes scheduler's backend appropriately[25][26]. Compared to cloud servers, edge servers exhibit higher network heterogeneity. Edge computing has more strict latency requirements than cloud computing. As an additional measure, the default scheduler in Kubernetes employs the queuing approach but with some drawbacks[27].

VI. LITERATURE REVIEW

This study emphasizes edge-aware routing in Kubernetes for multi-region deployments by analyzing several studies. Table I provides a summary of the findings:

Douch et al. (2022) provide an in-depth analysis of Edge Computing and the technologies that support it in an organized and organized manner. They begin with a basic definition of EC, then go on to describe its designs and how it has progressed from Cloudlets to Multi-Access Edge Computing. The next step is to review current research on the fundamental components of an EC system, such as controlling resources, offloading computing, managing data, and networks [28].

Marchese and Tomarchio (2022) suggest enhancing the standard Kubernetes scheduler with a network-aware plugin that can handle the unpredictable network circumstances on cloud-edge clusters and the communication needs of microservices during runtime. Additionally, a unique de-scheduler is suggested that routinely checks the network condition at runtime and the traffic flowing between microservices and evicts. Pods from cluster nodes in the event that more intelligent placement choices can be made. Our scheduling and rescheduling techniques have been tested in a test bed environment using a prototype microservices-based application[29].

Nguyen, Phan and Kim (2022) propose a Resource Adaptive Proxy (RAP) to enhance Kubernetes (K8s) load balancing for edge computing. Unlike kube-proxy, which distributes requests equally, RAP dynamically monitors pod resources and network conditions to optimize request routing. It prioritizes local processing and redirects traffic to available nodes when necessary, reducing latency and improving throughput. Experimental results show RAP outperforms K8s' default load balancing in edge environments[2].

Raj et al. (2021) examine how load balancing enhances edge computing, which brings computation and storage closer to data sources to reduce latency, improve response time, and optimize bandwidth. As cloud applications replace data center-based implementations and IoT devices proliferate, edge computing becomes crucial for efficient deployment[30].

Kodakandla (2021) explore the challenges of applying Kubernetes (K8s) to edge computing, including resource constraints, network instability, latency requirements, and security risks. It highlights solutions such as lightweight K8s distributions (K3s, MicroK8s), edge-aware scheduling (Kube Edge), and networking advancements (service meshes, 5G integration). AI-driven workload optimization and predictive analytics are examined for improved performance. A comparative analysis demonstrates reduced latency and optimized resource usage. Real-world applications in healthcare, retail, manufacturing, and smart cities showcase K8s' transformative potential at the edge. This study provides a roadmap for advancing K8s adoption in edge environments[31].

Lee et al. (2022) suggest using EdgeX over Kubernetes, a product-grade container orchestration solution that runs EdgeX Foundry over Kubernetes, allowing for remote service deployment and autoscaling to application services. Experiment findings show that the suggested platform enhances system throughput and service quality with autoscaling and real-time monitoring, and it makes the system more manageable by allowing application services to be deployed remotely [32].

Table I presents a comparative analysis of various research studies focusing on Edge Computing and Kubernetes-based Solutions.

TABLE I. SUMMARY OF RELATED WORK FOR ANALYSIS OF KUBERNETES-BASED EDGE SOLUTIONS

Reference	**Research Focus**	**Framework**	**Analysis**	**Key Advantages**	**Limitations**	**Future Work**
Douch et al. (2022)	Evolution of Edge Computing and its enabling technologies	Cloudlets, Multi-Access Edge Computing (MEC), Virtualization, Resource Management	Literature survey and architectural assessment	Comprehensive overview of EC evolution and key components	Lacks experimental validation and real-world case studies	Implementation of proposed EC strategies in practical environments
Marchese and Tomarchio (2022)	Network-aware scheduling for Kubernetes in cloud-edge clusters	Kubernetes, Microservices, Custom Scheduler & Descheduler	Testbed deployment and empirical evaluation	Improved pod placement efficiency, reduced network overhead	High dependency on Kubernetes architecture, limited scalability tests	Optimization for large-scale cloud-edge deployments

Nguyen, Phan and Kim (2022)	Adaptive load balancing in Kubernetes for edge environments	Kubernetes (K8s), Resource Adaptive Proxy (RAP), Network Optimization	Experimental comparison with default kube-proxy	Lower latency, higher throughput, improved request routing	Requires additional overhead for continuous resource monitoring	Integration with AI-based decision-making for adaptive load balancing
P. Raj et al. (2021)	Load balancing techniques in edge computing	Edge Computing, IoT, Load Balancing Algorithms	Theoretical and case study analysis	Optimized bandwidth usage, reduced latency	Lacks practical implementation and real-world validation	Implementation of AI-driven load-balancing strategies in edge computing
Kodakandla (2021)	Challenges of Kubernetes in edge computing and proposed solutions	Kubernetes (K3s, MicroK8s, KubeEdge), AI-driven optimization, 5G	Comparative study of Kubernetes distributions and real-world applications	Improved resource efficiency, reduced deployment complexity	Limited scalability and security concerns	AI-driven predictive analytics for edge workload optimization
Lee et al. (2022)	EdgeX over Kubernetes for remote service orchestration	EdgeX Foundry, Kubernetes, Autoscaling, Real-Time Monitoring	Experimental evaluation of system performance and service orchestration	Enhanced manageability, increased throughput, automated service deployment	Potential performance bottlenecks under high workload	Enhancing system robustness for industrial IoT applications

VII. CONCLUSION AND FUTURE WORK

Kubernetes networking is essential for secure and efficient microservice communication in cloud-native environments. Edge-aware DNS routing in Kubernetes for multi-region deployments significantly enhances service availability reduces latency and optimizes resource utilization by intelligently directing traffic based on geographical location, real-time network conditions, and load distribution. By integrating edge computing principles, this approach ensures efficient data processing closer to end-users, thereby improving fault tolerance, regulatory compliance, and overall system resilience. The role of Core DNS in managing service discovery, DNS resolution, and traffic distribution across clusters is crucial for seamless cloud-edge integration, particularly in dynamic 4G and 5G environments. Despite these advantages, several challenges persist, including infrastructure complexity, security risks, data consistency issues, and resource constraints, necessitating robust orchestration and network management strategies.

Future research should focus on AI-driven adaptive DNS routing to further enhance decision-making based on predictive network analytics, as well as topology-aware container scheduling to improve workload distribution. Additionally, integrating real-time monitoring with machine learning techniques can provide dynamic optimization of DNS traffic flows, ensuring greater resilience and efficiency in multi-cloud and edge environments. Further advancements in secure DNS mechanisms, such as DNS-over-HTTPS (DoH) and DNSSEC, will be essential to mitigating potential security vulnerabilities. Moreover, investigating the impact of emerging technologies like 6G networks and decentralized computing paradigms on edge-aware DNS routing can provide new opportunities for optimizing cloud-edge orchestration. By addressing these challenges and exploring innovative routing techniques, future work can drive the evolution of Kubernetes networking toward more adaptive, intelligent, and resilient multi-region deployments.

REFERENCES

[1] M. Gopalsamy, S. Cyber, and S. Specialist, "Advanced Cybersecurity in Cloud Via Employing AI Techniques for Effective Intrusion Detection," *IJRAR*, vol. 8, no. 1, pp. 187–192, 2021.

[2] Q. M. Nguyen, L. A. Phan, and T. Kim, "Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy," *Sensors*, 2022, doi: 10.3390/s22082869.

[3] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "@inproceedings{al2019container, title={Container orchestration engines: A thorough functional and performance comparison}, author={Al Jawarneh, Isam Mashhour and Bellavista, Paolo and Bosi, Filippo and Foschini, Luca and Martuscelli, Giuseppe and Montanar," *IEEE Netw.*, vol. 32, no. 1, pp. 102–111, 2018.

[4] I. M. Al Jawarneh *et al.*, "Container Orchestration Engines: A Thorough Functional and Performance Comparison," in *IEEE International Conference on Communications*, 2019. doi: 10.1109/ICC.2019.8762053.

[5] S. I. Shamim, J. A. Gibson, P. Morrison, and A. Rahman, "Benefits, Challenges, and Research Topics: A Multi-vocal Literature Review of Kubernetes," *Softw. Eng.*, 2022.

[6] M. Iorio, F. Risso, A. Palesandro, L. Camiciotti, and A. Manzalini, "Computing Without Borders : The Way Towards Liquid Computing," pp. 1–18, 2022.

[7] N. Hu, S. Yin, S. Su, X. Jia, Q. Xiang, and H. Liu, "Blockzone: A decentralized and trustworthy data plane for DNS," *Comput. Mater. Contin.*, 2020, doi: 10.32604/cmc.2020.010949.

[8] P. Herbert Raj, "An edge dns global server load balancing for load balancing in edge computing," *Lect. Notes Data Eng. Commun. Technol.*, 2021, doi: 10.1007/978-981-16-0965-7_57.

[9] E. Karaarslan and E. Adiguzel, "Blockchain based DNS and PKI solutions," *IEEE Commun. Stand. Mag.*, 2018, doi: 10.1109/MCOMSTD.2018.1800023.

[10] J. Arundel and J. Domingus, *Cloud Native DevOps with Kubernetes*. 2019.

[11] F. Minna, A. Blaise, F. Rebecchi, B. Chandrasekaran, and F. Massacci, "Understanding the Security Implications of Kubernetes Networking," *IEEE Secur. Priv.*, vol. PP, pp. 2–12, 2021, doi: 10.1109/MSEC.2021.3094726.

[12] V. S. Thokala, "Utilizing Docker Containers for Reproducible Builds and Scalable Web Application Deployments," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 6, pp. 661–668, 2021, doi: https://doi.org/10.14741/ijcet/v.11.6.10.

[13] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge

computing and big data processing," *J. Supercomput.*, 2019, doi: 10.1007/s11227-018-2701-2.

[14] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases, and Future Directions," *IEEE Communications Surveys and Tutorials*. 2017. doi: 10.1109/COMST.2017.2717482.

[15] C. Chang, S. N. Srirama, and R. Buyya, "Internet of things (IOT) and new computing paradigms," in *Fog and Edge Computing: Principles and Paradigms*, 2019. doi: 10.1002/9781119525080.ch1.

[16] H. H. Pang and K. L. Tan, "Authenticating query results in edge computing," in *Proceedings - International Conference on Data Engineering*, 2004. doi: 10.1109/ICDE.2004.1320027.

[17] J. Taheri and S. Deng, "Edge computing: Models, technologies and applications," *Edge Comput.*, no. March 2021, pp. 1–453, 2020, doi: 10.1049/PBPC033E.

[18] T. Integration, "A Survey of Multi-Access Edge Computing in 5G and Beyond :," vol. 4, 2020, doi: 10.1109/ACCESS.2020.3001277.

[19] H. Tabatabaee Malazi *et al.*, "Dynamic Service Placement in Multi-Access Edge Computing: A Systematic Literature Review," *IEEE Access*, vol. 10, pp. 32639–32688, 2022, doi: 10.1109/ACCESS.2022.3160738.

[20] H. Rahimi, Y. Picaud, K. D. Singh, G. Madhusudan, S. Costanzo, and O. Boissier, "Design and Simulation of a Hybrid Architecture for Edge Computing in 5G and beyond," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1213–1224, 2021, doi: 10.1109/TC.2021.3066579.

[21] L. Hou, M. A. Gregory, and S. Li, "A Survey of Multi-Access Edge Computing and Vehicular Networking," *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3224032.

[22] D. N. Molokomme, A. J. Onumanyi, and A. M. Abu-Mahfouz, "Edge Intelligence in Smart Grids: A Survey on Architectures, Offloading Models, Cyber Security Measures, and Challenges," *J. Sens. Actuator Networks*, vol. 11, no. 3, 2022, doi: 10.3390/jsan11030047.

[23] S. Böhm and G. Wirtz, "Towards Orchestration of Cloud-Edge Architectures with Kubernetes," 2021. doi: 10.1007/978-3-031-06371-8_14.

[24] R. Bruschi, F. Davoli, G. Lamanna, C. Lombardo, S. Mangialardi, and J. F. Pajo, "Enabling edge computing deployment in 4G and beyond," *Proc. 2020 IEEE Conf. Netw. Softwarization Bridg. Gap Between AI Netw. Softwarization, NetSoft 2020*, pp. 237–241, 2020, doi: 10.1109/NetSoft48620.2020.9165384.

[25] K. Nakanishi, F. Suzuki, S. Ohzahata, R. Yamamoto, and T. Kato, "A Container-based Content Delivery Method for Edge Cloud over Wide Area Network," 2020, pp. 568–573. doi: 10.1109/ICOIN48656.2020.9016481.

[26] P. Kayal, "Kubernetes in Fog Computing: Feasibility Demonstration, Limitations and Improvement Scope : Invited Paper," 2020, pp. 1–6. doi: 10.1109/WF-IoT48130.2020.9221340.

[27] S. Böhm and G. Wirtz, "EAI Endorsed Transactions Cloud-Edge Orchestration for Smart Cities : A Review of Kubernetes-based Orchestration Architectures," vol. 6, no. 18, 2022.

[28] S. Douch, M. R. Abid, K. Zine-Dine, D. Bouzidi, and D. Benhaddou, "Edge Computing Technology Enablers: A Systematic Lecture Study," *IEEE Access*, vol. 10, no. July, pp. 69264–69302, 2022, doi: 10.1109/ACCESS.2022.3183634.

[29] A. Marchese and O. Tomarchio, "Network-Aware Container Placement in Cloud-Edge Kubernetes Clusters," in *Proceedings - 22nd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2022*, 2022. doi: 10.1109/CCGrid54584.2022.00102.

[30] P. Raj, "An Edge DNS Global Server Load Balancing for Load Balancing in Edge Computing," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 66, 2021, pp. 735–742.

doi: 10.1007/978-981-16-0965-7_57.

[31] N. Kodakandla, “Optimizing Kubernetes for Edge Computing: Challenges and Innovative Solutions,” *Iconic Res. Eng. Journals*, vol. 4, no. 10, pp. 210–221, 2021.

[32] S. Lee, L.-A. Phan, D.-H. Park, S. Kim, and T. Kim, “EdgeX over Kubernetes: Enabling Container Orchestration in EdgeX,” *Appl. Sci.*, vol. 12, no. 1, 2022, doi: 10.3390/app12010140.