Analysis and Evaluation of Kubernetes based Multi-Cluster Service Discovery Service Mesh Solutions

Robert O'Brien

Supervisor: Dr. Siobhan Drohan

MSc Enterprise Software Systems

January 2021

Contents

List of Figures

List of Tables

Acronyms

Plagiarism Declaration

I confirm that this assignment is my work, that it is not copied from any other person's work (whether published or unpublished). I am aware that any user of material from other works (including paraphrase) without citation will be treated as plagiarism. The sources of any material quoted or referenced are provided and acknowledged.

Robert O'Brien

1 Introduction

MSAs (MSA) have seen a rise in popularity due to containerised orchestration platforms such as K8s (K8s). They facilitate creating smaller, loosely coupled services that can be released independently of each other, accelerating development cycles and increasing flexibility. Container Orchestration Platforms such as the K8s manage and orchestrate containerised applications, which are application packaged into individual, lightweight runtime environments containing everything needed to run a microservice. Load Balancing, automated scaling, and service discovery (SD) are just some of the features provided by container orchestration platforms.

The advantages offered by container orchestration platforms are not without cost. The greater the number of services running in a distributed system, the more complex it becomes. Microservices are often dependent on each other and need to communicate to work as part of a more extensive, coherent system. K8s clusters are traditionally single cluster setups, but as many companies and organisations move their microservice-based applications to container orchestration platforms, multi-cluster designs are becoming more common. SD is an intrinsic part of modern MSAs and not just container orchestration platforms. Interservice communication within modern MSAs have had to evolve to accommodate the ever-growing number of services that make up today's applications. Complexity is hard to scale, even more so when dependent services to span multiple clusters. Service Meshes (SM) help reduce this complexity by controlling the communication within MSAs.

This study analyses how current multi-cluster SD SM solutions solve this problem, as many take a novel approach. A variety of solutions will then be evaluated using the following categories: configuration complexity, health-checks, load balancing and security.

2 Background

2.1 Multi-Cluster

A single K8s instance is termed a cluster. The term multi-cluster refers to two or more independent K8s clusters used to form a larger connected system of clusters that communicate over a network. K8s multi-cluster setups are becoming more popular. Support for multiple cloud providers and hybrid environments has increased, primarily to allow for increased isolation for specific services and require high performance and fault tolerance levels. One of the drawbacks of multiple K8s clusters is the absence of a feature allowing services in different clusters to discover each other over a network. As multi-cluster SD is not a native K8s feature, many third-party solutions have stepped in to solve this problem using various architectural approaches and discovery mechanisms. Figure 1 below shows a basic multi-cluster federation architecture:

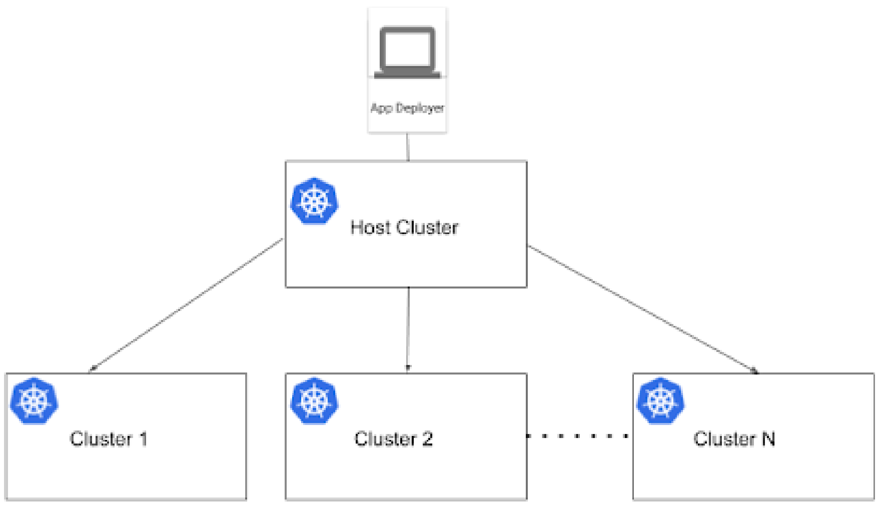

Figure 1: Example of a multi-cluster federation architecture (Suryanarayanan, 2020)

2.2 Service Discovery

K8s is the process by which a service exposed from a cluster is made available for DNS requests from clients (Suryanarayanan, 2020). The clients in a multi-cluster setup can consist of services located on different clusters, a single service deployed on multiple clusters or external services outside of a cluster that consumes the service. Microservices within a K8s cluster are deployed in what are known as pods. Each pod is assigned an internal IP address, which allows communication between pods within the same cluster. Still, due to a pods' ephemeral nature (they can be created, scaled, destroyed and updated during their lifecycle), a pod's IP address is a dynamic property. To account for this, K8s uses an additional resource known as a service. A service is an abstraction above a grouping of pods that receives all networking requests, and load balances the requests between the pods associated with the service. As the services' IP address is static (unless configured otherwise), the cluster's multiple services can communicate. While this is sufficient for a single cluster setup, clusters in a multi-cluster setup have no way of knowing what services are exposed outside its boundary. Figure 2 below displays an example of how the Submariners Lighthouse project facilitates cross-cluster SD:

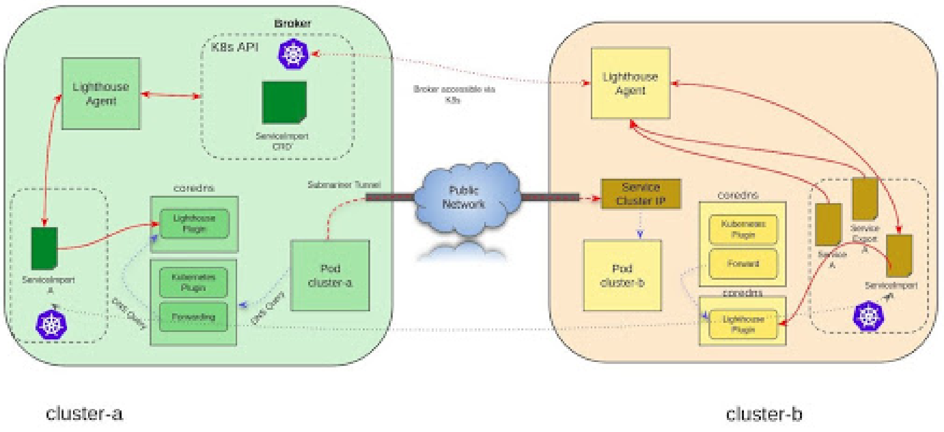

Figure 2: Multi-cluster service discovery architecture with Submariner and Lighthouse (Farrell, 2020)

2.3 Service Mesh

A service mesh (SM) is a configurable, low-latency infrastructure layer designed to handle a high volume of network-based interprocess communication among application infrastructure services using application programming interfaces (Smith and Garrett, 2020). The most common SM design pattern is the sidecar design pattern. In this pattern, a proxy service (commonly known as a sidecar) is deployed alongside each service to act as a gateway for requests bound for the service. As all networking requests travel through the sidecar proxy, networking concerns such as authentication, observability, routing, and security are handled by the acrshortsm, allowing for a more resilient networking design. Many of today's SMs offer multi-cluster SD. Figure 3 below displays an example of the sidecar design pattern within a SM:

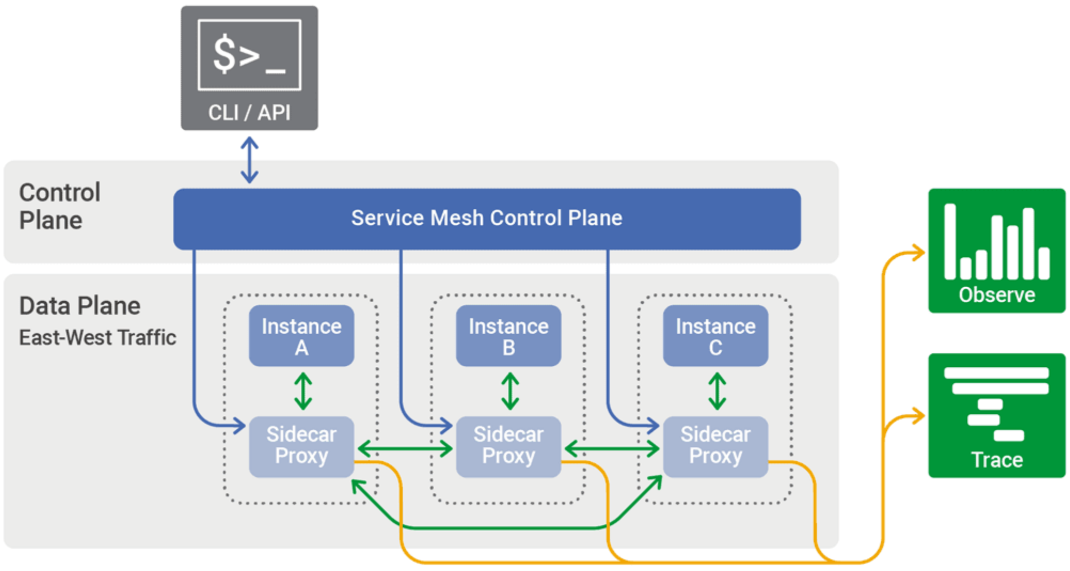

Figure 3: Generic service mesh topology (Smith and Garrett, 2018)

3 Scope

This study's primary goal is to analyse multi-cluster SM SD solutions to determine what design architectures and discovery mechanism each solution employs to overcome the challenges associated with multi-cluster SD. Each solution will then be evaluated under the following categories: configuration complexity, health-checks, load balancing and security. The evaluation process will use a Proof of Concept environment consisting of two or more K8s clusters to simulate a multi-cluster setup. The solutions under evaluation study must be open-source and in a mature state of development. Hybrid or multi-cloud K8s configurations are outside of the scope of the study. Time permitting, the study will be extended to compare the SD architecture of the current solutions with the proposed solution in the K8s Enhancement Proposal (KEP) related to this problem.

4 Research Problem

There is currently no standard way to connect or even think about K8s services beyond the cluster boundary. Still, we increasingly see users deploy applications across multiple clusters designed to work in concert (Olmsted-Thompson, 2020). While many third-party SM solutions exist to alleviate this problem, the design approaches and method each takes differs significantly. It is not always obvious which solution is the most suitable for each specific use case or performs best in specific areas. This study will analyse each solution's design architecture and implementation approach before evaluating each under four chosen categories which make up important decision factors associated with SD: configuration complexity, service level health-checks, service level load balancing and inter-service security. Native support in the K8s community for multi-cluster SD is scheduled in the form of a K8s KEP. As of writing, the KEP is in an Alpha development stage and no target K8s release version or time frame exists as to when this feature will be made generally available.

5 Research Questions

1. What design architectures and discovery mechanisms do current open-source Service Mesh solutions employ to facilitate multi-cluster service discovery?
2. How do each of the solutions compare in terms of cross cluster service discovery for:
   (a) Configuration Complexity
   (b) Health-Checks
   (c) Load Balancing
   (d) Security

   when evaluated on a multi-cluster K8s setup?

6 Preliminary Literature Review

Li et al. (2019) state that there are very few research works in service mesh in academia. Literature in the context of multi-cluster SD using a SM is lacking, even with the growing number of mature tools available. This shortage is attributed to SMs being developed initially for single cluster setups, with multi-cluster support being added to the majority of solutions in more recent releases.

6.1 Analysis

(Malki and Zdun, 2019) outline 14 qualitative design decisions used to guide microservice architectural designs in a service mesh environment. These design decisions include Health-Checks, Managed Cross-Service Communication, Load-Balancing, Security, and the SM Expansion decision. the authors discuss a variety of SM architectural designs associated with each decision. These include the service proxy the centralised control plane and distributed control plane designs. The paper doesn't offer guidance on what design architecture may suit a multi-cluster environment, or what designs might be best suited for such a setup, even though the design decisions put forward by the authors include a SD decision.

Discovery mechanisms for cloud-based microservice SD solutions are analysed in a paper by Braubach et al. (2018) which looks at the implications of SD within systems containing a large number of services ($10^6 - 10^9$). The paper looks at HashiCorps Consul, Netflix's Eureka, and Apache Zookeeper, popular non-SM solutions. While Consul and Netflix's Eureka use a service registry based architecture to employ SD, Zookeeper uses a distributed key-value store. The authors state that both Consul and Eureka contain problematic design decisions for megascale systems, with one of the reasons being query processing not using indexed data structure. Hashicorp (the creators of Consul) also offer one of the most popular SM solutions that also uses a service registry to facilitate SD, as do many of the available solutions. As these are non-SM solutions, it's not clear if these problem carry over to SMs, but if the same mechanism (service registries) is used, it is possible. Tato (2019) also focuses the Consul, Eureka and Zookeeper and Consul SD solutions, noting how Zookeeper is a centralised solution which does not scale well in multi-datacentre solutions, with the opposite being true for Consul and Eureka. A gap in the literature exists in the context of SD mechanisms, as much of the existing literature is focused on their non-SM counterparts.

6.2 Evaluation

SD challenges in distributed systems are well documented. Stubbs et al. (2015) acknowledge these associated challenges by evaluating several then-current solutions against a custom project called Serfnode, a project developed by the paper's authors. The paper is from 2015, before the widespread adoption of K8s or container orchestration platforms (K8s dates from 2014) and before cloud-native Service Meshes were introduced into the K8s ecosystem. As the author's state in the paper, Serfnode is optimised for use within a single data centre and can be deployed into an existing MSA to provide SD mechanisms, monitoring, and self-healing capabilities. Many of today's SM solutions such as Istio, Consul and Linkerd solve many of the same challenges that Serfnode was developed for, while also offering many additional features. SM solutions have also significantly evolved since 2015, and scope exists to evaluate the solutions both used in the paper as they exist today, and solutions created since the paper's publication.

The security challenges associated with SMs are outlined by Hahn et al. (2020). The paper details the steps taken to evaluate different SM solutions in terms of security features and configurations. As security protocols and configuration parameters can differ between various solutions, testing using default configurations makes sense to correctly evaluate each solution. The authors acknowledge this by conducting the evaluation under an idealised scenario which compares the tool's default offerings. Hahn et al. (2020) is another paper which the evaluates the security of SMs, with the authors testing the default security configuration of each solution. As security protocols and configuration parameters can differ between various solutions, testing when using the default configuration gives the best view of what the solution offers. a‘Li et al. (2019) performs a state of art review of SM solutions in a number of categories which include project activeness, major advantage, critical limitation and overall maturity. The paper also discusses fundamental features on a SD which includes SD and load balancing (which includes status health checks). Even though both of the above mentioned papers use qualitative metrics to evaluate a specific aspect or feature of the SD, there is a no literature available that evaluates SMs in the context of SD (using qualitative or quantitative methods).

As outlined in the previous analysis section, Malki and Zdun (2019) provide several design decisions that help the microservice architectural design process. SM health-checks are discussed with the authors outlining the different types of health-checks that can be performed: simple, periodic, complex or none. Similarity, the load-balancing decision describes the different types of load-balancing options available. As neither of these two decisions are discussed in the context of a multi-cluster set-up, an opportunity exists to carry out on evaluation to on both service-level load-balancing and health-checking as the literature is lacking in both areas in the context of SMs.

7 Contribution to Research Knowledge Anticipated

The research will present a comprehensive analysis of how each SM solution approaches and implements multi-cluster SD from an architectural design perspective and also under the following categories:

1. Configuration Complexity
2. Health-Checks
3. Load Balancing
4. Security

As K8s does not currently support SD across multiple clusters, end-users must evaluate the available solutions to determine which best fits their use case and meets their requirements. The available solutions take various approaches to the problem, thus evaluating and understanding how each performs and compare in the above categories will enable users to make a more informed decision.

8 Description of the Experimental Design/-Validation Methodology

8.1 Research Approach

Qualitative research questions will form the basis of the evaluation of each of the categories, as outlined in Table 1. The qualitative methods will entail observations based on testing on a local workstation and the study and review of existing documentation for each solution. The result of the evaluation will be documented in the form of a table that will provide answers to each of the evaluation questions for each of the solutions.

Category	**Metric**
Configuration Complexity	What configuration is required to implement service discovery? How is the solution managed and maintained?
Health-Checks	What service level health check methods does the solution support?
Load Balancing	What cross cluster inter-service load balancing algorithms and configurations does the solution support?
Security	What inter-service security policies and protocols does the solution support by default? Does the solution support authentication protocols?

Table 1: Evaluation Categories and Metrics

8.2 Research Method

The analysis stage of the research focuses on the multi-cluster SD architectural design approach of each SM. Many of the prospective solutions have accompanying documentation detailing the approach taken. Each solution will also be deployed to a local K8s cluster, which will further allow for a detailed view of the each of the solution's components.

The evaluation phase will require a minimum of two K8s clusters to simulate configuring a multi-cluster environment. Existing tools, such as MiniKube and KIND (Kubernetes In Docker), allow the creation of multiple clusters on a local workstation. Test clusters use an identical configuration on the same local workstation to ensure a consistent, reproducible testing environment. The deployment and the solutions onto the test cluster will be followed by an evaluation process under the categories outlined in the prior section.

9 Resources Required

9.1 Hardware

A single laptop capable of running multiple K8s SIG (Special Interest Group) local test clusters (Kind/Minikube).

9.2 Software

- The SMs and under evaluation
- Third-party testing tools for the performance evaluation
- A microservice-based K8s application that can be used in the evaluation stage

10 Main Milestones Anticipated

10.1 Multi-cluster configuration set-up

January/February 2021

Local K8s multi-cluster set-ups will be investigated to determine which is the most suitable for this study. Once determined, a consistent, reproducible testing environment will be created in which to test each solution. A suitable MSA application will be sourced and deployed onto the multi-cluster set up to facilitate the evaluation stage.

10.2 Selection and analysis of Service Mesh solutions

February/March 2021

Multiple SMs will be selected using the solution evaluation criteria set out in the scope section. Each solution's architectural and design approach will then be analysed to discover how each solution approaches the SD problem.

10.3 Evaluation of Service Mesh solutions

March/April 2021

An analysis of the selected solutions on a multi-cluster environment will occur under the following categories outlined in *RQ2*.

10.4 Interim report submission and review

May 2021

An Initial draft of the dissertation containing the work on the previously anticipated milestones.

References

Braubach, L., Jander, K. and Pokahr, A. (2018), ‘A novel distributed registry approach for efficient and resilient service discovery in megascale distributed systems?’, *Comput. Sci. Inf. Syst.* **15**(3), 751–774. [online] Available at: `http://www.doiserbia.nb.rs/img/doi/1820-0214/2018/1820-02141800030B.pdf` [Accessed 4 January 2020].

Farrell, D. (2020), ‘Submariner lighthouse architecture’. [online] Available at: `https://submariner.io/images/lighthouse/architecture.png` [Accessed 27 December 2020].

Hahn, D., Davidson, D. and Bardas, A. (2020), ‘Mismesh: Security issues and challenges in service meshes’. [online] Available at: `https://arxiv.org/pdf/2010.11079.pdf` [Accessed 28 December 2020].

Li, W., Lemieux, Y., Gao, J., Zhao, Z. and Han, Y. (2019), Service mesh: Challenges, state of the art, and future research opportunities, *in* ‘2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)’, pp. 122–1225. [online] Available at:`https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8705911` [Accessed 7 January 2020].

Malki, A. E. and Zdun, U. (2019), Guiding architectural decision making on service mesh based microservice architectures, *in* ‘ECSA’. [online] Available at:`http://eprints.cs.univie.ac.at/6073/1/paper.pdf` [Accessed 7 January 2020].

Olmsted-Thompson (2020), ‘Kubernetes/Enhancements’. [online] Available at: `https://github.com/kubernetes/enhancements/tree/master/keps/sig-multicluster/1645-multi-cluster-services-api#summary` [Accessed 26 December 2020].

Smith, F. and Garrett, O. (2018), ‘What is a service mesh? - nginx’. [online] Available at: `https://www.nginx.com/wp-content/`

uploads/2019/02/service-mesh-generic-topology.png [Accessed 26 December 2020].

Stubbs, J., Moreira, W. and Dooley, R. (2015), 'Distributed systems of microservices using docker and serfnode'. [online] Available at: https://ieeexplore.ieee.org/document/7217926/ [Accessed 27 December 2020].

Suryanarayanan, A. (2020), 'Multicluster service discovery in openshift (part 1)'. [online] Available at: https://www.openshift.com/hubfs/Google%20Drive%20Integration/Multicluster%20Service%20Discovery%20in%20OpenShift-2.png [Accessed 26 December 2020].

Tato, G. (2019), Lazy and locality-aware building blocks for fog middleware : a service discovery use case, PhD thesis, Rennes University. [online] Available at: https://tel.archives-ouvertes.fr/tel-02570516/document, [Accessed 4 January 2021].