SIDECAR PROXY-BASED APPROACH TO CHANGE MICROSERVICE COMMUNICATION STRATEGY AT RUNTIME

Y.M.V.M. Yapa

229412H

Degree of Master of Science

Department of Software Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2024

SIDECAR PROXY-BASED APPROACH TO CHANGE MICROSERVICE COMMUNICATION STRATEGY AT RUNTIME

Y.M.V.M. Yapa

229412H

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science in Software Engineering

Department of Software Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2024

DECLARATION

I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 2024/06/28

The above candidate has carried out research for the Master's thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Prof. Indika Perera

Signature of the Supervisor: Date: 2024/06/28

DEDICATION

I would like to dedicate this to my family who supported me through the whole journey of my higher education.

ACKNOWLEDGMENT

I want to thank Prof. Indika Perera for the valuable guidance provided to finalize the thesis. I want to thank the Department of Computer Science staff and my friends who provided materials and guidance to build a proper thesis. I would also like to thank my family for encouraging me to complete this.

ABSTRACT

Microservice architecture is becoming popular for most application development in modern, rapid, requirement-changing systems. Although the ideal situation in this architecture is to develop loosely coupled systems with possibly zero interservice communication, it is practically unavoidable in most cases. Selecting the right communication strategy is one of the hardest decisions [1][2][3][4][23][24], and most often, if that decision is made incorrectly during the initial phase, it becomes harder to change afterward. Things are getting even more difficult because of the emerging new techniques in microservice communication. Selecting the right solution depends on many factors, such as personal preferences, tradeoff analysis, and experience. It is difficult to decide whether the correct communication strategy will be chosen until the application runs in the production environment with a real load. Most solutions and suggestions come from experience or after a failure. Since the development effort is high, developing different solutions and then comparing them to select the right solution is impossible. No proper prototyping techniques are available to evaluate such strategies without spending too much time and cost. Most applications might not need the flexibility of changing the communication strategies at the runtime, but when needed, it becomes a very hard challenge to get success. These things are sometimes never addressed due to cost factors, even if they are identified in the first place. Hence, a proper solution decoupled from the application logic is required.

A solution is developed to address the adding dynamic behaviors to communication in this research. This solution tries isolating the communication strategy from the microservice by introducing a sidecar proxy object and the core microservice module. The sidecar proxy is a pattern that involves deploying a helper service (proxy) alongside each microservice instance in a distributed application to manage network communications, security, and observability. This proxy handles cross-cutting concerns such as service discovery, load balancing, and traffic routing, allowing the main microservice to focus solely on its core logic.

Most applications use static application configurations to implement a particular communication strategy. Another major outcome of this method is the introduction of the ability to change the communication strategy at runtime using a dynamic application configuration management system. This allows the configurations associated with the communication strategies to change at runtime.

It is also necessary to evaluate this approach's suitability using characteristics like security and performance. This proposed solution is not coupled to any programming language or framework. It is also a hybrid approach that can only be applied to the selected applications. This can also be used to develop prototype applications quickly and evaluate the available solutions quickly.

Keywords: Microservice architecture, Sidecar proxy, Runtime software evolution

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
REST	Representational State Transfer
RPC	Remote Procedure Call
IPC	Inter-Process Communication

LIST OF APPENDICES