

LB/TH/24/2025

TH5870

**OPTIMIZED STRATEGY IN CLOUD-NATIVE
ENVIRONMENT FOR INTER-SERVICE
COMMUNICATION IN MICROSERVICES**

L.D.S.B Weerasinghe

208092X

Doctor of Philosophy

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

May 2025

OPTIMIZED STRATEGY IN CLOUD-NATIVE ENVIRONMENT FOR INTER-SERVICE COMMUNICATION IN MICROSERVICES

L.D.S.B Weerasinghe

208092X

Dissertation submitted in partial fulfillment of the requirements for the degree
Doctor of Philosophy

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

May 2025

DECLARATION

I declare that this is my own work, and this 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: 20/05/2025

The above candidate has carried out research for the PhD 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: 20.05.2025

Dedicated to my family for their constant love, support, and encouragement
throughout my journey.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor, Prof. Indika Perera, for his invaluable support and guidance throughout my research journey. His commitment to my work, coupled with his constructive feedback and insightful criticisms, has been instrumental in shaping my research skills and helping me navigate this complex field. Prof. Indika's support in publishing my work in reputed conferences and journals has been a cornerstone in advancing my academic and professional growth.

I am deeply appreciative of the time and effort dedicated by my examiners, Dr. Amal Shehan Perera and Dr. Kutila Gunasekera, whose constructive feedback during my progress reviews has been invaluable. Their insights have helped refine and strengthen the contributions of this thesis.

I would also like to extend my gratitude to Axiata Digital Labs for the flexibility that allowed me to take academic responsibilities. This support has been invaluable in my ability to pursue my research with dedication.

My heartfelt thanks go to the Head of the Department of Computer Science and Engineering, the Dean of the Faculty of Engineering, and the Dean of the Faculty of Postgraduate Studies for their steadfast support and guidance through the administrative aspects of my PhD journey. I also wish to extend my appreciation to the staff of the postgraduate division and the establishment division for their assistance and friendliness in handling the procedural aspects of my studies.

Finally, I am deeply grateful to my loving mother and father, whose unwavering belief in my potential and constant encouragement have been the foundation of my success. Their understanding and patience throughout my PhD journey, even when my commitments took me away from family obligations, have been a source of strength. To my wife, my heartfelt thanks for her steadfast support, encouragement, and understanding as I dedicated long hours to this work. Her empathy and sacrifices have been invaluable in helping me persevere through the challenges of this journey. This accomplishment would not have been possible without the unconditional love and support of my family.

ABSTRACT

Microservices architecture has become the most popular architecture for building scalable and flexible applications due to its inherent advantages over monolithic and Service-Oriented Architecture (SOA) systems. It allows developers to scale individual components independently, improve maintainability through smaller codebases, and increase resilience by isolating service failures. Additionally, microservices are well-suited for cloud-native environments, supporting containerization and orchestration platforms. However, transforming monolithic systems into microservice-based systems introduces challenges with inter-service communication, leading to performance bottlenecks and increased latency due to the decentralized and distributed nature of the architecture. This research addresses the above challenge by proposing a novel communication model for cloud-native environments, focusing on optimizing inter-service communication in microservice architecture. The proposed strategy operates on top of the TCP network layer rather than the application layer, leveraging TCP streams but using a request-response-based communication model. All distributed systems typically rely on TCP-based communication over a network. To reduce the overhead of opening and closing socket connections, the proposed strategy persists the TCP connection when a microservice starts and maintains it until the service goes down. This solution reduces network resource consumption by serializing messages into binary form, thereby enhancing response times and application throughput. When sending messages over the TCP connection, we have established a method for ensuring exact-once delivery, improving the reliability and efficiency of message transfer. The proposed solution was implemented within a microservice architecture and deployed in a cloud-native environment to assess its effectiveness for cloud-native applications. The solution has been evaluated against the traditional HTTP-based communication method in the same microservice-architected environment and has shown improved performance in terms of latency and overall throughput. This research offers a robust communication model for microservices that addresses performance challenges while maintaining compatibility with cloud-native concepts. The proposed approach is a step toward optimizing microservices communication, enabling developers to build scalable, efficient, and resilient applications in distributed environments.

Keywords: Microservices, Inter-service communication, Performance

TABLE OF CONTENTS

Declaration	i
Acknowledgement.....	iii
Abstract	iv
Table of Contents	v
List of Figures	viii
List of Tables.....	x
List of Abbreviations.....	xi
Chapter 1	1
Introduction	1
1.1 Software Architecture.....	2
1.1.1 Monolithic architecture behavior	4
1.1.2 Service oriented architecture behavior.....	5
1.1.3 Decentralized and distributed architecture.....	7
1.2 Motivation	9
1.3 Problem Definition	10
1.4 Objectives	11
1.5 Research Questions	11
1.6 Publication Contribution	14
1.7 Thesis Structure	17
Chapter 2	19
Literature review	19
2.1 Introduction	19
2.2 Monolithic architecture	19
2.3 Service Oriented Architecture	24
2.4 Monolithic to Microservices Architecture.....	31
2.4.1 Migration Issues and Problems	36
2.4.2 Migration Challenges	37
2.5 Microservice Architecture	38
2.5.1 Challenges & Difficulties in Microservice Architecture	45

2.6	Inter Service Communication.....	45
2.6.1	HTTP/HTTPS	47
2.6.2	gRPC	48
2.6.3	WebSocket	48
2.6.4	Deep Dive into Factors Influencing Performance Metrics in Inter-Service Communication	49
2.7	Software Quality Attributes.....	52
2.8	Cloud Computing	53
2.8.1	Public Clouds	56
2.8.2	Private Clouds	57
2.8.3	Hybrid Clouds	58
2.8.4	Community Clouds	59
2.8.5	Service Models of Cloud Computing.....	59
2.8.6	Cloud Computing Trends.....	61
2.9	Concluding Remarks	63
Chapter 3		64
Methodology		64
3.1	Introduction	64
3.2	Evolutionary Analysis of Microservice Architecture.....	64
3.3	Key Considerations for Optimizing Inter-Service Communication.....	67
3.3.1	Technique to Reduce Protocol Overhead and Improve the Network Latency.....	67
3.3.2	Techniques to Minimize Programming Overhead to Users.....	68
3.3.3	Ensuring Cloud Native Compatibility.....	69
3.3.4	Software Quality Attributes Preservation	69
3.4	Sustainable Framework	70
3.5	Solution Architecture	72
3.6	Technology Selection	73
3.7	Concluding Remarks	76
Chapter 4		77
Proposed Solution		77
4.1	Introduction	77

4.2	High-level Architecture	78
4.3	High-level Component Architecture	79
4.4	Low-level Design	81
4.4.1	Request Flow	82
4.4.2	Response Flow	85
4.5	Achievement of Quality Attributes	87
4.6	Cloud Architecture	89
4.7	Concluding Remarks	93
Chapter 5		94
Results and evaluation.....		94
5.1	Introduction	94
5.2	Inter-service Communication Protocol Comparison	94
5.2.1	Controlled Throughput & Payload Size	97
5.2.2	Controlled Requests & Payload Size	99
5.3	Proposed Solution Evaluation	101
5.3.1	Proposed Solution Testing Scenario A	103
5.3.2	Proposed Solution Testing Scenario B.....	104
5.4	Proposed Solution Cloud Native Suitability	106
5.5	Evaluating the Number of Segments for Inter-service Communication ...	111
5.6	Reference System	115
5.7	Concluding Remarks	118
Chapter 6		119
Conclusion		119
6.1	Introduction	119
6.2	Problem and Contribution	119
6.3	Concluding Findings for Research Questions	121
6.4	Achievement of the Objectives	124
6.5	Limitations.....	126
6.6	Future Work	127
6.7	Summary	129
References		131

LIST OF FIGURES

Figure	Description	Page
Figure 2.1	Monolithic architecture	20
Figure 2.2	Proposed monolithic architecture for testing	23
Figure 2.3	Service oriented architecture	25
Figure 2.4	Extended SOA	28
Figure 2.5	Microservice architecture	32
Figure 2.6	Monolithic to microservice decomposition using static and dynamic analysis	35
Figure 2.7	Framework for monolithic to microservice conversion	35
Figure 2.8	Saga pattern	43
Figure 2.9	API composition pattern	43
Figure 2.10	Command side replica pattern	44
Figure 2.11	CQRS pattern	44
Figure 2.12	OSI Layer	51
Figure 3.1	PRISMA model analysis	65
Figure 3.2	Microservice publication distribution	66
Figure 3.3	Sustainable framework for the improvement of inter service communication	70
Figure 3.4	High-level solution architecture	72
Figure 4.1	High-level architecture	78
Figure 4.2	High-level component architecture	80
Figure 4.3	Request flow	82
Figure 4.4	REST controller implementation	82
Figure 4.5	External API implementation	83
Figure 4.6	Additional configurations for proposed solution	83
Figure 4.7	Initial stream subscription creation	83
Figure 4.8	Publish message to stream	84
Figure 4.9	How clients used the proposed solution to send the messages	85
Figure 4.10	Response flow	85
Figure 4.11	Consume message from stream	86

Figure 4.12	Structure of event object	86
Figure 4.13	Cloud architecture	89
Figure 4.14	High-level deployment architecture	90
Figure 4.15	High-level cloud deployment architecture	92
Figure 5.1	Protocol comparison deployment architecture	95
Figure 5.2	Protocol comparison testing architecture	96
Figure 5.3	Protocol-wise turnaround time variation	98
Figure 5.4	Impact on response time for TPS in various protocols	99
Figure 5.5	Proposed solution testing architecture	102
Figure 5.6	Proposed solution turnaround time comparison	104
Figure 5.7	Proposed solution TPS, response time and turnaround time comparison	105
Figure 5.8	Cloud-native testing architecture	107
Figure 5.9	HTTP vs proposed solution turnaround time comparison in cloud environment	108
Figure 5.10	Variation of response time and turnaround time with CPU	109
Figure 5.11	Variation of response time and turnaround time with memory	110
Figure 5.12	Turnaround time variation with scaling	110
Figure 5.13	Deployment test architecture for evaluating the segments of communication paths	112
Figure 5.14	Inter service communication paths	113
Figure 5.15	Response time variation with number of microservices	113
Figure 5.16	Inter-service communication variation with microservices	114
Figure 5.17	Response time variation with communication paths	115
Figure 5.18	Reference system deployment architecture	116
Figure 5.19	Reference system response time variation	117

LIST OF TABLES

Table	Description	Page
TABLE 2.1:	Categories of SOA benefits	29
TABLE 5.1:	Protocol comparison testing scenarios	98
TABLE 5.2:	Proposed Solution Test cases	103
TABLE 5.3:	Cloud-native test cases	107

LIST OF ABBREVIATIONS

Abbreviation	Description
OOA	Object-Oriented Architecture
SOA	Service Oriented Architecture
ESB	Enterprise Service Bus
IaaS	Infrastructure as a Service
SaaS	Software as a Service
PaaS	Platform as a Service
FaaS	Function as a Service
TPS	Transactions per second
API	Application Programming Interfaces
JMS	Java Message Service
XML	Extensible Markup Language
WSDL	Web Services Description Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
RPC	Remote Procedure Call
TCP	Transmission Control Protocol
IP	Internet Protocol
REST	Representational State Transfer
RESP	Redis Serialization Protocol
LLD	Low Level Diagram
VM	Virtual Machine
JVM	Java Virtual Machine
GCP	Google Cloud Platform
GKE	Google Kubernetes Engine
CI	Continues Integration
CD	Continues Delivery