“Analysis and Development of protocol to enhanced security orchestration in Cloud Computing SLA”

Thesis

Submitted for the award of

Degree of Doctor of Philosophy

Computer Science and Engineering

By

Shikha Singh

Enrollment No: MUIT0116038021

Under the Co-Supervision of **Dr. Himanshu Pandey**

Under the Supervision of **Dr. Ajay Kumar Bharti**

Under the Maharishi School of Computer Science and Engineering

Session 2016

Maharishi University of Information Technology

Sitapur Road, P.O. Maharishi Vidya Mandir
Lucknow, 226013

Supervisor Certificate

This is Certify that Mrs. Shikha Singh has completed the necessary academic term and the work presented by her is a faithful record of bonafide original work under the guidance and supervision of **Dr. Ajay Kumar Bharti, Professor, School of Computer Science and Engineering, Maharishi of Information Technology, Lucknow, India**. She has work on **ANALYSIS AND DEVELOPMENT OF PROTOCOL TO ENHANCED SECURITY ORCHESTRATION IN CLOUD COMPUTING SLA.** No part of this thesis has been submitted by the candidate for the award of any other degree or diploma in this and any other university around the globe.

Under the Co-Supervision of

Dr. Himanshu Pandey
Assistant Professor
Computer Science and Engineering
Lucknow University
Lucknow ,U.P.,India

Under the Supervision of

Dr. Ajay Kumar Bharti
Professor
School of Computer Science and Engineering
Maharishi University of Information Technology
Lucknow,U.P. 226013,India

Date:
Place: Lucknow

Declaration by the Scholar

I hereby declare that the work presented in this thesis entitled "**Analysis and Development of protocol to enhanced security orchestration in Cloud Computing SLA**" in fulfillment of the requirements for the award of Degree of Doctor of Philosophy, submitted in the Maharishi School of **Computer Science and Engineering Department**, Maharishi University of Information Technology, Lucknow is an authentic record of my own research work carried out under the supervision of **Dr. Ajay Kumar Bharti** and/or co-supervision of **Dr. Himanshu Pandey** I also declare that the work embodied in the present thesis-

i) is my original work and has not been copied from any journal/ thesis/ book; and

ii) has not been submitted by me for any other Degree or Diploma of any University/ Institution.

Shikha Singh
Signature of the Scholar

Table of Contents

LIST OF FIGURES

List of Tables

LIST OF ABBREVIATIONS

Abbreviation	Description
AES	Advance Encryption Standard
API	Application Programming Interface
APM	Application Performance Management
APTs	Advanced Persistent Threats
ART	Abort Resume Terminate
ASD	Automatic Service Deployment
ASPD	Application Service Provider Device
AWS	Amazon Web Services
CA	Certificate Authority
CAPIs	Cloud Application Programming Interfaces
CASViD	Cloud Application SLA Violation Detection Architecture
CC	Cloud Computing
CC	Canonical Correlation
CIA	Confidentiality Integrity Availability
CFSM	Communicating Finite State Machines
CM	Credential Manager
CMC	Container's Migration Cost
COBIT	Control Objectives for Information Technologies
CPU	Central Processing Unit
CRC	Cyclic Redundancy Checks
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
CSS	Cloud Service Subscriber

DDoS	Distributed Denial of Service
DoS	Denial of Service
DEF	Diffe Hellman Encryption
DES	Data Encryption Standard
DeSVi	Detecting SLA Violations
DSLs	Domain Specific Language
EC_2	Elastic Cloud Compute
ECDHE	Ephemeral Elliptic Curve DHE
FEP	Fair Exchange Protocol
FIFO	First in First Out
FSM	Finite State Machine
GCD	Google Cloud Platform
GSLAM	Generic SLA Management
GUI	Graphical User Interface
HIPAA	Health Insurance Portability and Accountability Act
HTTP	Hyper Text Transfer Protocol
HTTP	Hyper Text Transfer Protocol Secure
IaaS	Infrastructure as a Service
ICSS	International Classification of Standardization
IDEA	International Data Encryption Algorithm
IDM	Identity Management
IOPS	I/O Operation Per Second
IT	Information Technology
ITSM	Information Technology Service Management
KPI	Key Performance Indicator
LA	Late Acceptance
LDAP	Lightweight Directory Access Protocol
LOM2HiS	Low Level Metrics to High Level SLAs
LSA	Late Simulation Annealing
LTL	Linear Temporal Logic
MB	Meta Broker

MF	Message Forward
MFA	Multi- Factor Authentication
MITC	Man in the Cloud
MITM	Man in the Middle
MMC	Machine Move Cost
MN	Meta Negotiator
NIST	National Institute of Standards and Technology
NSG	Network Security Group
OS	Operating System
PaaS	Platform as a Service
PDA	Personal Digital Assistant
PDM	Performance degradation Migration
POC	Planning and Optimization
POP3	Post Office Protocol
PROMELA	Process or Protocol Meta Language
PE	Protocol Engine
PKI	Public Key Infrastructure
P2P	Point to Point
QoS	Quality of Service
ROI	Return on Investment
RCA	Root Cause Analysis
RC4	River Cipher
RSA	River Shamer Adleman
RDP	Remote Desktop Protocol
SaaS	Software as a Service
SA	Simulation Annealing
SBNP	Simple Bilateral Negotiation Protocol
SIEM	Security Incident and Event Management
SLA	Service Level Agreement
SLA@SOI	Service Level Agreement @ Service Oriented Infrastructure

SLAC	Service Level Agreement Cloud
SLAV	SLA -Violations
SLOs	Service Level Objectives
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Security Orchestration Automation Response
SRV	SLA-based Resource Virtualization
SSH	Secure Shell
TEA	Tiny Encryption Algorithm
TOPOSCA	Topology and Orchestration Specification for Cloud Applications
TOs	Trusted Organizations
TS	Tabu Search
TSDB	Time Series Data Base
TTP	Trusted Third Party
UMIN	Unique Message Identification Number
VM	Virtual Machine
VPC	Virtual Private Cloud
WSDL	Web Service Description Language
WSLA	Web Service Level Agreement
WSOL	Web Service Offerings Service
XML	Extensible Markup Language

List of Symbols

Symbols	Nomenclature
(∧)	Conjunction
(∨)	Disjunction
(¬)	Negation
(→)	Implication
(∪)	Until

Acknowledgement

I would like to express my deepest gratitude to Dr. Ajay Kumar Bharti, my research supervisor and co-supervisor Dr. Himanshu Pandey for their unwavering support, invaluable guidance and continuous encouragement throughout entire duration of my doctoral studies. Their expertise, patience and mentorship have been instrumental in shaping the direction and quality of their research.

I am also immensely thankful to the members of my thesis committee. Their insightful comments, constructive feedback and rigorous evaluation have significantly contributed to the improvement of this thesis.

I extend my sincere appreciation to Maharishi University of Information and Technology, Lucknow for providing me with the conducive research environment and the necessary resources to pursue my doctoral studies. The infrastructure facilities and academics opportunities offered by university have been crucial to the successful completion of this research.

I would like to acknowledge the support and encouragement of my supervisor and co supervisor who have stood by me during the challenging journey. Their intellectual discussions and sharing of ideas and moral support have been invaluable and I am grateful for their in my life.

I am deeply indebted to the participation of my study, whose involvement and contribution made their research possible. Their willingness to share their insights and experiences has enriched the finding and added depth to the study's conclusion.

Furthermore, I would like to express my gratitude to Maharishi University of Information and Technology, Lucknow for granting me to access their resources and allowing me to gather the necessary data for this research. Their cooperation and support were essential to the successful completion of this study.

Shikha Singh

Abstract

Cloud computing is becoming more important since it facilitates the transition of existing services from client-server architectures to cloud computing, in addition to allowing digital trading platforms. Providers and customers all across the globe have benefited from the technological and economic advancements made in this sector. Virtualization of services is becoming more important in a variety of industries, and digital marketplaces are being utilized for much more than the trade of physical items. Recent years have seen a rise in the frequency and severity of service disruptions for many prominent cloud providers, including Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform. Human mistake, ineffective change control, and more lately, targeted cyber-attacks like distributed denial of service (DDoS) are all causes for these disruptions, which affect the whole supply chain and result in the temporary suspension of multiple widely used online services. These web-based solutions provide cloud service subscribers (CSS) with a platform that may be used to access and make use of various services in the cloud, such as computing, storing data, and communicating with other users. CSP and CSS sign a legally binding agreement, such as a Service Level Agreement, before any cloud service is implemented (SLA). A failure guarantee and service scope are included in the SLA. Either side to the SLA, often the CSP, has the option to terminate the agreement at any time.

The use of SLAs as tools to establish and sustain trust in a society where interdependence between services is becoming more and more important is discussed. By identifying and resolving pertinent research issues, the thesis proposes general approaches to automate SLA lifecycle management. The techniques allow for adaptability in a dynamic corporate environment and may be localized using policy-based restrictions. Business models, services, and delivery technology are distinct ideas that may be delicately woven together by SLAs, according to a conceptual vision that arises from this effort. The message of this thesis is supported by experimental evaluations, which show that using SLAs as the cornerstones of market innovation and infrastructure regulation does, in fact, offer win-win scenarios for both cloud clients and cloud providers. It is also difficult to enforce SLA and prevent potential disputes and inappropriate behaviour during the exchange of cloud-based services because our proposed architectures utilize automated and engage with hardcoded procedures and verification mechanisms, incorporating a diverse set of trusted third parties and authorities.

Keywords: Cloud Computing, Security Orchestration, Service Level Agreement, Generic Service Level Agreement, Security Orchestration Automation and Response, Fair Exchange Protocol, Low Level Metrics to High Level SLA, Detecting SLA Violation, Cloud Application SLA Violation Detection Architecture, Simple Bilateral Negotiation Protocol ,Service Level Agreement @ Service Oriented Infrastructure ,Security Incident and Event Management ,Infrastructure as a Service, Platform as a Service , Software as a Service, Trusted Third Party , Unique Message Identification Number, Average Response Time, Protocol Engine ,Cloud Service Provider ,Cloud Service Subscriber

CHAPTER 1

Introduction

1.0 Overview

The term “the cloud” now represent a rising IT paradigm that provides convenient, instant network access to a shared collection of programmable resources and advanced services. The term "cloud orchestration" refers to the use of programming technology to control the dependencies and interactions between various cloud-based tasks. In order to achieve a common objective, it orchestrates a series of independent automated processes into a unified workflow that is then subject to controlled access and strict policy enforcement. The most common uses for cloud orchestration are to purchase and assign storage space, provide servers, manage networking, spin up virtual machines, and unlock specialized software. This is made possible by the three primary, interconnected aspects of cloud orchestration: service, workload, and resource orchestration. Permission checks, which are essential for ensuring security and compliance, may be built into an orchestration platform. Because orchestration coordinates many automated processes, automation is a subset of orchestration. In general, automation aims to orchestrate processes such that a single job may be repeated quickly and with minimum human interaction.[1]

Among its two primary contracts with clients, the service level agreement stands out as particularly important. The standard terms and conditions for working with a service provider are often laid forth in a master services agreement. Most service providers will include the SLA as a reference in their master services agreement.The SLA supplements the other service contract with further detail on the services to be delivered and the measures to be used to evaluate their efficacy. With the knowledge that a shorter time to market increases the likelihood of success, businesses are increasingly turning to orchestration to standardize and improve regular, repeatable processes in order to guarantee the correct, faster deployment of software.Whenever there is a need to cut out unnecessary steps from a process and those steps are repeatable, the process can be optimized. The IT budget may be better allocated toward new and innovative initiatives thanks to Security Orchestration's cost-cutting benefits.Bringing about less conflict amongst different groups, Productivity growth and enhancement, creating uniformity in practice.[2]

1.2 Cloud Computing

Growing demands for IT services necessitate a more robust IT framework and infrastructure. IT service providers must meet the increasing needs of their consumers, but they are challenged by the need to grow their systems with limited resources and quick turnaround. The development of the cloud computing architecture arose in response to the need to meet the needs of businesses and to satisfy commercial interests. IT infrastructure for certain services is hosted in the cloud and made available to clients on demand. When discussing cloud computing, the term "cloud provider" is used to refer to the business that offers the service to end users. Cloud customers, on the other hand, are the businesses that really use cloud services. It's true that the idea of "the cloud" has been around for a while, but its popularity and importance are only going to grow over the next decade. Several factors, such as the decreased price and energy usage of the pooled computer resources, have contributed to its growing popularity (servers, software, storage, and networking). This not only allows for the efficient utilization of IT resources, but also provides greater adaptability for the instantaneous expansion of new infrastructures. [3]

Cloud computing poses dangers and security problems for businesses, much like conventional computer systems. Problems with privileged user access, meeting legal and regulatory requirements, separating data, backing it up, recovering it, investigating wrongdoing, and the cloud provider's long-term survival are all examples of risks and security problems. Due to these difficulties, cloud users must put in place procedures to monitor and enhance the safety of their cloud-based data. Information security metrics are one of the options offered to cloud customers for keeping tabs on and, in turn, enhancing the safety of their data when it comes to the assets that are managed in the cloud.

Information security metrics can be implemented by a cloud client via a Service Level Agreement since cloud resources are provided on demand (SLA). The primary legal foundation for managing and regulating the services provided is the SLA,(Service Level Agreement) serves as contractual arrangement between the cloud service provider and consumer. Metrics based on SLAs are used to compare the quality of service provided by a cloud provider to its clients. However, present cloud SLAs tend to assess performance rather than security. The GoGrid SLA is an example of such a contracting agreement. Information security concerns are one barrier to cloud computing adoption at the company. However, there are still cases of cloud service companies ignoring the issue. [4]

In the IT world, "the cloud" refers to a novel approach of sharing server and other computing resources. Services are the typical delivery mechanism for such resources. Cloud computing service architecture is divided into three models of services, with four distinct deployment options. The services are distinguished from the conventional computer environment by their own core properties. As a result, we will discuss the features, services, and deployment models of cloud computing in this section.

1.2.1 Architecture of Cloud Computing

Here, we lay out a high-level framework for cloud computing, including many models for providing cloud services. Collaboration, adaptability, scalability, and availability are all boosted by the cloud, and the possibility of reduced costs thanks to streamlined and efficient processing power is another perk. Cloud computing, in this context, refers to the deployment of a shared virtualized computing, networking, and storage environment to support multiple users and applications (CSA Security Guidance, 2009).Employing a dynamic allocation and consumption paradigm resembling on demand utilities, these parts may be swiftly organized, provided, installed, and decommissioned. As a result of the benefits they provide in terms of integration, provisioning, orchestration, mobility, and scalability, virtualization technologies are widely utilized in conjunction with cloud services.[5]

Although "cloud" implies a distancing of resources from their ties to and position inside the infrastructure providing them, some interpretations of cloud go to extremes by over emphasizing or artificially limiting cloud's many qualities. One common reason is to try to impress upon others how important it is when it is not. Some definitions appear to be missing crucial context, such as the assumption that a service must use the Internet serves as a transport, a web browser act as an access mode and resources are consistently shared a multi-tenant scenario beyond the traditional “perimeter."

There is a lot of uncertainty about how cloud adoption in relation to conventional network and information security practices will affect organizational, operational, and technological strategies because of the cloud's similarities and differences from existing models, which is to be expected given the cloud's abstracted evolution of technology. While some see cloud computing as a revolutionary leap forward, others see it as the logical progression of technology, economics, and culture. The truth lies in the middle, someplace. [6]

Numerous models have been proposed to approach cloud from various stakeholder perspectives, including those of academics, architects, engineers, developers, managers, and consumers. In this chapter, we will examine an architecture that was designed with the specifics of IT network deployment and service provisioning in mind.

There are five main tenets of cloud services that show how they are similar to and different from more conventional forms of computing. In this regard, we may identify the following:

1.2.1.1 Abstraction of infrastructure

The delivery of services involves isolating computer, network and storage infrastructure resources from applications and data .The data provider's application or service no longer has to worry about where their data is being processed, sent, or stored; these factors are effectively hidden. Infrastructure resources are often pooled whether a shared or dedicated tenancy strategy is employed. These types of partitions are often provided by highly virtualized chipsets and operating systems, or by the greater degree of abstraction made possible by heavily modified file systems, operating systems, or communication protocols.

1.2.1.2 Resource democratization

Enabling the pooling of resources and granting authorized access to them through standardized method is what we call "resource democratization," and it's made possible by the abstraction of infrastructure. This includes physical infrastructure, software applications, and data. [7]

1.2.1.3 Service-oriented architecture

By separating infrastructure from applications and data, a services-oriented architecture is created in which resources may be accessed and exploited in a consistent manner thanks to well-defined and loosely-coupled resource democratization . In this setup, the service itself takes precedence over the underlying infrastructure.

1.2.1.4 Elasticity/dynamism

Self-service models that expand to as-needed capacity are made possible by the combination of the implementing of cloud provisioning model that offers the on-demand availability, high level of automation, virtualization and reliable, high-speed connectivity. It is possible that a greater level of service may be offered and more resources employed effectively if they are pooled together.

1.2.1.5 Utility model of consumption and allocation

Dynamic resource allocation may be achieved by combining the cloud's abstracted, democratized, service-oriented, and elastic qualities with stringent automation, orchestration, provisioning, and self-service. Since resource usage is now easily measurable, a metered system of utility pricing and utilization may be put into place. This allows for better cost control and predictability, as well as greater scalability and economies of scale.

1.2.2 Cloud computing characteristics

These five characteristics delineate the key attributes of cloud computing:

1.2.2.1 On-demand self-service

Cloud customers can demand computing capabilities such as network storage.

1.2.2.2 Broad network access

Users have the ability to access the cloud resources through the diverse interfaces that are accessible via the internet. (e.g.: laptop, PDA).

1.2.2.3 Resource pooling

As part of a multi-tenancy model, a cloud service provider pools its computing resources to serve a number of different customers.

1.2.2.4 Rapid elasticity

Capabilities can be rapidly and flexibly acquired as per the specific requirements. Customers have the impression that they have unlimited access to the features and may buy them whenever they choose. [8]

1.2.2.5 Measured service

Utilizing metering capabilities tailored to each service type, cloud systems can automatically regulate and optimize resource consumption. Bandwidth, data storage, and so forth. Both the cloud service provider and the end user may see exactly how many resources were used and for what purpose.

1.2.3 Cloud computing services

The three primary classifications of cloud services are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Customers of the cloud can choose from a variety of services in each of these three groups. Customers of a cloud service have the option of selecting the precise services they need. In the following paragraphs, a condensed explanation of each of the three offerings will be provided. Cloud service architecture is depicted in Figure 1.1 below.

1.2.3.1 Software as a Service (SaaS)

With Software as a service (SaaS), user can access the required software without need for downloading and installing it on their personal machine. The SaaS provider stores the software and makes it accessible online. Software as a service applications are multi-tenant applications, meaning they are used by more than one client at a time. However, each customer's applications must be considered conceptually distinct. When using a SaaS model, the provider bears the responsibility and safeguarding their clients' data. Word processors and content delivery networks serve as examples are two popular types of SaaS software. Businesses like Google and Salesforce.com provide SaaS.[9]

1.2.3.2 Platforms as a Service (PaaS)

PaaS stands for Platform as a Service, wherein a cloud provider provides a development environment suitable for running applications built by the service's end users. Using an API (Application Programming Interface), the platform can be altered from afar. The development tools, deployment platform, and configuration management are all part of the platform service. This means customers can get their applications up and running without having to learn complex administration techniques. Moreover, customers don't need to install any tools on their computers in order to construct and deploy their web applications. [10]

PaaS providers, like SaaS providers, are tasked with protecting the rented services. Two layers of code are responsible for PaaS security:

- The safety of the PaaS infrastructure itself. As an example, a runtime engine may be built into a PaaS.
- Client applications deployed to a PaaS environment must be protected.

Consequently, the safety of both the platform and client applications is the responsibility of the PaaS provider. Microsoft Azure Google App Engine are just a couple of instances of companies offering such services.

1.2.3.3 Infrastructure as a Service (IaaS)

Infrastructure as a service provides network access to virtual machines and other forms of abstracted hardware and operating system. Customers may avoid keeping up with infrastructure updates and take use of cutting-edge tools by subscribing to an IaaS provider. When it comes to IaaS, consumers have more control over security than they do with SaaS and PaaS. GoGrid, Flexiscale, and Amazon are just some examples of companies that provide this service.

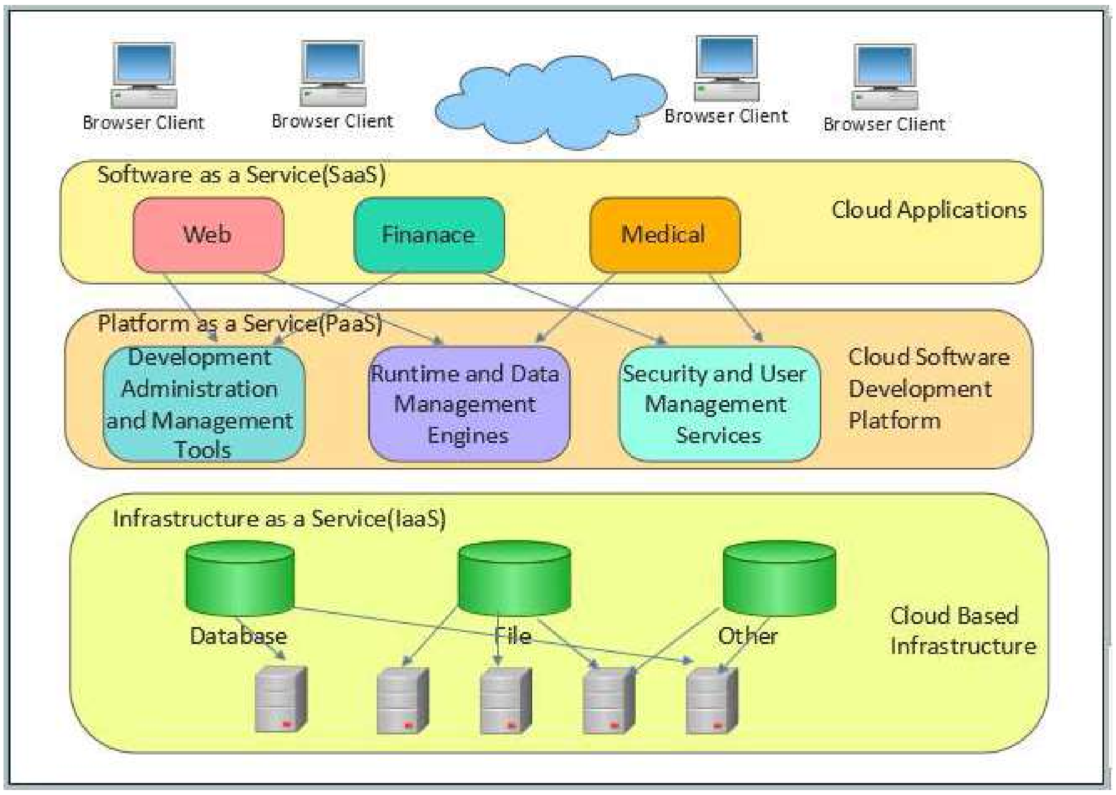

Browser Client
Browser Client
Browser Client
Browser Client
Software as a Service(SaaS)
Web
Finanace
Medical
Cloud Applications
Platform as a Service(PaaS)
Development Administration and Management Tools
Runtime and Data Management Engines
Security and User Management Services
Cloud Software Development Platform
Infrastructure as a Service(IaaS)
Database
File
Other
Cloud Based Infrastructure

Figure 1.1: Cloud services architecture

1.2.4 Cloud Deployment Model

The cloud may be hosted and delivered in a variety of ways, similar to how different services are implemented. The many cloud deployment models include:

1.2.4.1 Private cloud

Private clouds have cloud infrastructure set up, but only for one company to use. Organizations who plan to use this kind of cloud deployment can only ever have access to it. A private cloud may be administered either internally or by an external provider. Here, cloud servers could or might not be physically situated in the same country as the company.

1.2.4.2 Community cloud

Through the community cloud model, multiple entities work together to provide cloud services to a community with similar needs. Multiple entities work together to create and maintain this cloud infrastructure, which serves a community with similar needs. This can be managed internally or outsourced to a third party and it can be located either within or outside the physical premises of the organization.

1.2.4.3 Public cloud

The public cloud model allows the general public or a significant industry association to access and utilize the infrastructure provided by the cloud.A for-profit company controls cloud storage. Large industrial groups or the general public have access to this type of infrastructure. Organizations providing cloud services are responsible for and maintain these.[11]

1.2.4.4 Hybrid cloud

The cloud infrastructure in a hybrid architecture consists of many clouds working together (private, community, or public). Multiple clouds, both public and private, as well as community clouds, can coexist in this deployment setup. These composite clouds, formed by the combinations of different cloud type like "private and public "or "public and community, retain their distinctiveness while being interconnected through standardized or preconfigured technology, facilitating applications and data portability.

1.3 Security Issues In Cloud Computing

While CC has many advantages, including as power, flexibility, and simplicity of use, it also presents several security issues. While CC offers a new, more streamlined approach to access programmes and complete tasks, its widespread adoption faces a number of obstacles. Even a cursory examination of the literature in this area reveals a few problems. There are three of them, and they are: Concerns such as Service Level Agreements (SLAs), data to move, security, etc. Automatic updates are a feature of CC, so any changes made by an administrator to an app are immediately available to all users.

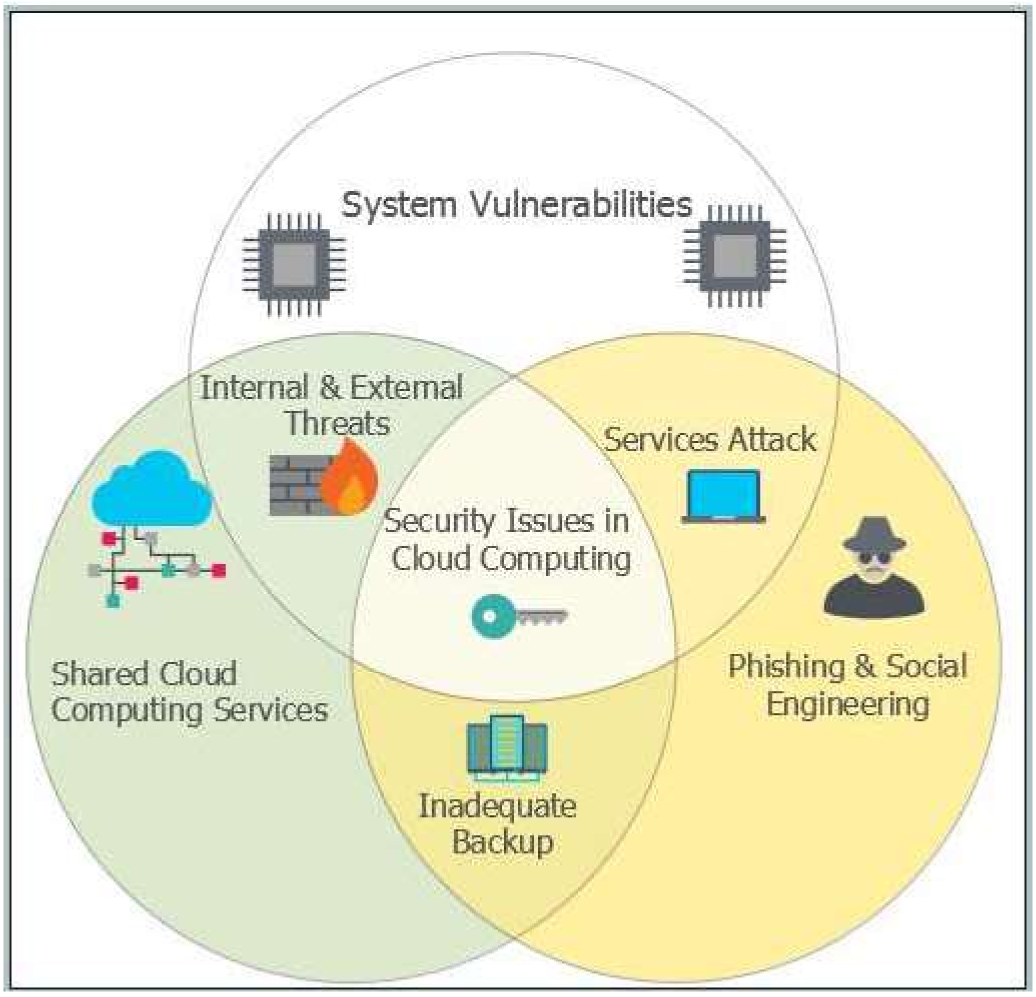

System Vulnerabilities
Internal & External Threats
Services Attack
Security Issues in Cloud Computing
Shared Cloud Computing Services
Phishing & Social Engineering
Inadequate Backup

Figure 1.2: Security Issues in the Cloud Computing

The growing importance of cloud computing has increased the need on businesses to implement security measures to safeguard the information stored in the cloud. Users have gained increased dependability, vast scalability, and an inexpensive cost of business development thanks to Cloud Computing services. As a result of advancements in cloud computing, large businesses now have the tools they need to not only fully automate their routine operations, but also provide employees with a convenient, web-based interface for managing their data and applications from anywhere with an internet connection.[12]

1.3.1 Cloud Computing Security threats

Cloud computing security risks encompass the following:

- In cloud computing, Lock-In Vendor occurs when a customer is overly reliant on one cloud provider, which might lead to inflexibility in the road. Vendor lock-in arises when transitioning between service providers or transferring ownership of online service becomes challenging.

- When a company makes the transition from on-premises IT infrastructure to cloud services without first developing and implementing appropriate governance rules and procedures, a problem known as "loss of governance" might arise.

- Compliance Any business that adheres to HIPAA, also known as the Health Insurance Portability and Accountability Act of 1996, has potential legal and ethical issues with the installation of cloud computing solutions to preserve the privacy and security of individual health information.

- After adopting Cloud Computing services, there is a risk of Supply Chain Failure because of problems with fulfilling orders. When clients place more orders online, it might strain their network and perhaps damage their reputation.

- Problems with cloud computing arise when essential technological requirements are not met, leading to disruptions in corporate operations. These requirements include sufficient processing infrastructure and sufficient user bandwidth.

- Malicious Another potential problem with cloud computing is that of "insiders," or individuals within the company whose negligence or other little mistakes might provide hackers and exploiters a way in.

- Known as "Shared Technology Cloud Services," this sort of cloud computing problem arises when an internet connection is shared and accessible to authorized parties there is an elevated risk of hackers and attackers potentially compromising sensitive data.

- Data and information stored or delivered using cloud computing technology must be encrypted using a complicated combination of cryptography and encryption techniques to prevent disclosure even if the information falls into the wrong hands. Weak or easily broken encryption might potentially cause problems in the cloud.

- Shared hosting is what we mean when we talk about "Multi-Tenancy," in which one instance of software represents several users and the server's resources are allocated accordingly.

- A Services Level Agreement (SLA) is a contract between cloud service providers that acknowledges when certain benchmarks have been met. Poor cloud computing services due to faulty Service Level Agreements (SLAs) can have a significant impact on a company's bottom line.

- DoS attacks, which are a prevalent danger for hosted online computing services, often include the injection of an overwhelming number of ping connection requests from a remote computer. In most cases of distributed denial of service, the system is brought to its knees by an overwhelming number of ping requests that exceed the cloud's capacity to handle. To protect against cyber risks and exploitations, it is important to correctly provision, administer, orchestrate, and monitor the system that provides cloud computing services to users so that it may be utilized, managed, and interacted with.

- The introduction of cloud computing raises concerns about data loss or leakage for enterprises, which necessitates the use of secure encryption and the assignment of appropriate protocols for all internet-based data transmissions.

- The term "data integrity" is used to describe how trustworthy and dependable data remains throughout time. Nobody who isn't allowed to access the data should see it, use it, or change it in any way. The dependable provision of the services such as PaaS, SaaS, and IaaS hinges on the precision of the data stored within the cloud.

- If a business just uses one system and doesn't have a backup server set up, it may suffer irreparable damage in the event of a natural disaster or a loss of availability, which might result in the loss of data and information that the company relies on.

- Companies that keep their data in the cloud face a severe risk when their backup fails, but this may be mitigated by having a few extra backup servers on standby.

- When discussing Cloud Computing, the term "Data Transfer Bottlenecks" is used to describe the limitations of the available bandwidth between the user's device and the cloud service provider. Due to the dynamic nature of cloud computing, it is easier to set up a wide-ranging corporate network, but this might introduce a latency issue if too many nodes in the network are involved in the data transmission.

- The objective of achieving interoperability in cloud computing to enhance the security of connections between public and private while facilitating a smoother exchange of data and communication between the two. The primary goal of interoperability is to allow cloud-based computing infrastructures to safely share and utilize data transmitted over the internet.

- Concerns about whether laws and regulations apply in the event of a lawsuit being filed in the cloud might be brought on by two primary factors: the physical location of the cloud service user and the nature of the application being delivered by the cloud service provider. When utilizing cloud computing services, businesses must ensure they are in compliance with all local laws and regulations concerning the deployment, sharing, and use of sensitive customer information.

- If a company wants to avoid paying out a fortune in damages because of a data breach, it needs to ensure that its data is secure in every possible environment, including the cloud. This means adhering to the laws of the relevant jurisdictions and implementing appropriate data encryption and protection measures.

- Licensing Risk may be more of an issue at the IAAS and PAAS levels of cloud computing since customers of these services do not have to worry about maintaining or controlling the underlying infrastructures, but they do have access to specific privileges inside the cloud-deployed services.

1.3.2 Purpose of Cloud Security Attacks and Exploitation

Cloud Malware Injection is a tactic utilized to compromise a user's cloud storage and gain unauthorized access to their data. The exploitation of cloud services, where system access or file control is obtained by an attacker, is frequently attributed to malicious individuals. A denial-of-service attack is an assault that is designed to make a cloud-based system unavailable to its intended users. Side-Channel Attacks are often used to gain access to a computer system or chip and harvest sensitive data. [13]

An application-layer wrapping attack involves the injection of a malicious element into a message structure with the intention of replacing an erroneous signature with a comparable signature that can be processed by the logic of the application. The main goal of a Man in the Cloud attack is to acquire access to the victim's account without obtaining the user's credentials, which are often compromised in such an attack. An insider attack is carried out with the purpose of gaining an unfair advantage, either for the perpetrator or their competitors. Payment or Access to a Service to get illegal access to a user's account and cloud sessions, a hijacking operation is undertaken. By exploiting Spectre and Meltdown, a process can have access to all of the memory in a cloud computing system by reading from random points inside the memory allocation of a programme. The goal of Advanced Persistent Threats (APTs) is to remain undetected on a network for an extended period of time in order to steal important information.

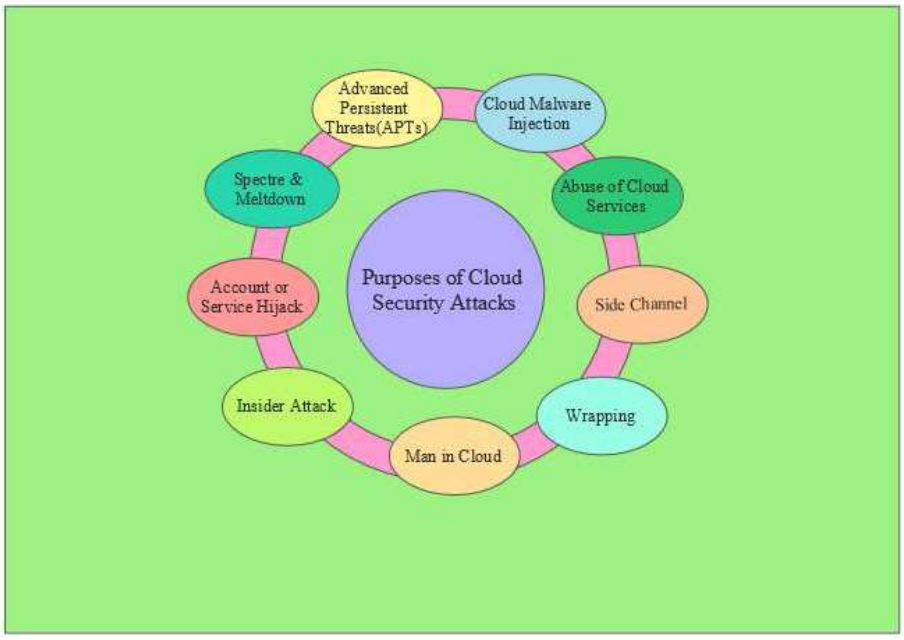


Figure 1.3: Purpose of Cloud Security Attack & Exploitation

1.4 Emerging Trends In Security And Privacy In Cloud Computing

The security, privacy and trust requirements of different domains, as well as their respective methods, interfaces, and semantics, can't be assumed to be the same in a single cloud computing system. Such a space might represent a service or any other form of infrastructure or application component that can function autonomously. Because they enable the construction and orchestration of services, service-oriented architectures are a naturally applicable technology for easing such cross-domain development. In order to construct a thorough policy-based management framework in cloud computing contexts, it is crucial to make use of current research on multi domain policy integration and the secure service composition. We discuss three critical issues related to cloud computing security and privacy that must be addressed before the technology can be widely used.[14]

1.4.1 Authentication and identity management

Customers may quickly and easily share their data with other online services by storing it in the cloud. To verify users and services using credentials and other attributes, an identity management (IDM) method might be used. Use of multiple identity tokens and identity negotiation techniques raises serious concerns for cloud-based identity management. Current password-based authentication systems have many inherent limitations and security flaws. Protecting sensitive user and system data necessitates a robust IDM solution. However, the impact of multitenant cloud environments on the privacy of identification information that requires further understanding. The problem of several jurisdictions might also make it difficult to implement effective safeguards. During the course of a user's interaction with a front-end service, that service may be responsible for maintaining the privacy of the user's identity from any back-end services with which it communicates. Providers in multi-tenant clouds must protect the privacy of their customers by keeping their identities and authentication details separate. Integration with other security components, like authentication and IDM, should be straightforward. Strong authentication and identity management procedures must be designed and developed before cloud computing can be implemented.

1.4.2 Access control and accounting

Due to the wide variety of services and varying access needs across domains, cloud computing systems require access control mechanism that offers precise and detailed control. Flexibility is a key requirement for access control services, ensuring they can adapt to diverse needs and scenarios of least privilege and to account for changing requirements for access depending on factors such as context, attribute, or credentials. Such access control systems may need the incorporation of intricate rules for privacy protection. An easy-to-use access control system is essential in the cloud, and so is the careful doling out of permissions. For good interoperability, cloud delivery models must be verified to offer generic access control interfaces, and in order to address across domain access challenges, the development of a policy neutral access control definition and enforcement framework is necessary. For this reason, it is crucial that researchers pay particular attention to the need for an easily auditable privacy-aware design for access control and accounting services.[15]

1.4.3 Trust management and policy integration

While numerous service providers work together in the cloud to offer a wide range of services, they may each use their own unique approach to security and privacy. As a result, we need to address the diversity in their approaches. The composition of numerous services by cloud service providers may be necessary to provide more complex application services. Therefore, measures are required to provide a safe interoperability process in which security breaches are efficiently monitored. Existing research has demonstrated that security infractions can arise readily during integration, even if specific domain regulations are checked. As a result, service providers must diligently over see their customers' access control rules to prevent any potential security issues from arising by integrating policies. Interactions between different service domains in cloud computing can be brief, intense, and influenced by specific service requirements. In order to effectively capture the broad range of trust characteristics and to keep up with the ever-changing trust and interaction/sharing needs, a trust framework has to be designed. Furthermore, difficulties managing challenges such as semantic heterogeneity, secure interoperability, and policy evolution become achievable through the policy integration responsibilities within the cloud. Given the dynamic nature of consumer behaviour, an adaptively supportive framework for policy integration is required to develop, negotiate, and sustain trust. Trust management framework design for wireless and P2P networks has received a lot of attention in the academic community. Yet, there is an immediate requirement for secure and trustworthy trust models in cloud computing settings. Due to the widespread use of cloud service delivery models and the inherent difficulties in ensuring compatibility across them, this is going to be an extremely difficult problem to solve.[16]

1.4.4 Secure service management

Entities operating as Cloud service providers and service integrators work together to tailor-make solutions for their customers in cloud computing environments. Independent service providers are able to orchestrate and interwork their services on the service integrator's platform, and together they can provide additional services that meet customers' security needs. While many cloud service providers make use of Web Services Description Language (WSDL) , traditional WSDL is inadequate to describe cloud computing services in their entirety. When searching for and composing services in the cloud, considerations like quality of service, cost, and service level agreements (SLAs) are essential. Problems like these needs fixing so that service descriptions and features can be introduced, interoperability can be maximized, integration can occur without compromising the fulfillment of service owner policies and service level agreements (SLAs) can be met. In essence, it is critical and urgent to develop a framework for automatically and systematically providing and composing services that take into account security and privacy concerns. [17]

1.4.5 Privacy and data protection

Protecting sensitive data like user identities, integration policies, and past financial transactions all present significant difficulties in the cloud. It's not uncommon for businesses to avoid putting sensitive information and programmes on servers located anywhere but in their own facilities. Data privacy is compromised when businesses move workloads to a common infrastructure. Cloud service providers are tasked with reassuring their clientele and ensuring their privacy by providing extensive transparency into their services. Each and every cloud security solution needs to have privacy protection built right in. The relevance of determining the original created of a piece of data, tracking modifications made to it, and understanding the process of such changes is steadily increasing as an associated concern. Tracing, auditing, and permissions based on past activity are just some of the potential applications for accumulated provenance data. Data provenance and privacy are two sides of the same coin that can be difficult to strike in the cloud, where traditional boundaries have been disregarded. Also, this is a major problem in the field of study.

1.4.6 Organizational security management

When businesses go to the cloud, they must rethink their current models to security management and the information security lifecycle. Shared governance, in particular, has the potential to become a major problem if it is not handled correctly. However, there is a risk that less collaboration will occur across client firms' various communities of interest if they adopt cloud computing. Concerns concerning the efficacy of business continuity and disaster recovery plans, as well as their quick response, might be exacerbated by reliance on third parties. The same is true when weighing the costs and benefits of various options. Emerging risks such as the potential for data leaking within multi-tenant clouds and concerns regarding resiliency, including the economics of providers and local calamities, are presented by a perimeter-less environment and must be considered by customers. To a similar extent, outsourcing data and operations to clouds greatly increases the likelihood of an insider attack. One tenant in a multi-tenant space may be the subject of a highly focused assault, which might have far-reaching consequences for all of the space's tenants .To enable customers to fully leverage the advantages of cloud computing, there is a need to reconsider existing models of life cycle modeling, risk analysis and management, penetration testing and service attestation must be rethought. [18]

Developing reliable security metrics that provide consistent and accurate data to aid in risk assessment has been a major challenge for the information security field. In order to guarantee the deployment and acceptance of secure clouds, we need to examine best practices and establish standards. These concerns call for a streamlined cyber insurance sector, but given cloud computing international scope, achieving this goal is no easy feat. Cloud-specific trends are important, but broader IT industry trends will also influence how cloud computing is practiced and how new services, architectures, and innovations are developed. Some examples of such tendencies include:

1.4.7 Increasing use of mobile devices

As numerous notebooks, PDAs, and mobile phones now incorporate functionalities that were prevalent in a desktop-based PCs merely decade ago the convergence of technology becomes evident , such as Internet connectivity and bespoke application capability, this trend is projected to continue, and laptop sales have already overtaken desktop PC sales. [19]

1.4.8 Hardware capability improvements

As the processing power of computers and the amount of storage space available in servers continues to grow, more complicated systems will be easily supported by the cloud, with their enhanced performance capabilities coming as standard.

1.4.9 Tackling complexity

Multiple technology providers have tried and failed to tackle this challenging problem. It is still challenging to develop, underutilized, and costly to operate IT architectures. IT systems that encompass diverse storage, servers, applications, storage and other components are capable of self-monitoring, self-healing, and self-configuring functionalities are more vital than ever in the era of cloud computing due to their massive scale.

1.4.10 Legislation and security

As larger firms start embracing the cloud computing paradigm, suppliers and providers will respond, but within the parameters and conditions established by those business. Since many people are still worried about how their personal information will be used and transmitted once it leaves their country, cloud service providers must maintain their commitment to investing time and resources energy into meeting the regulations necessary to do business in the industries of some of its most important customers. [20]

1.5. SLA in Cloud Computing

To meet the SLO, the client and provider must come to an agreement on the features and requirements of a SLA for the Cloud Computing service. When deciding on new services, it is crucial for clients to feel certain that their demands will be met in full.

Service level agreements (SLAs) must include and precisely describe the parameters of services, metrics, and functions in order to calculate the values of the service characteristics. To ensure the security of data and operations in a cloud context, some challenges must be acknowledged, addressed, and negotiated inside a SLA. Assurances of the safety of data, program, and hardware are among these features, as are audit ability for enforcing security, compensation for losses, credible certification, support during service disruptions, and so on (among others). [21]

Despite the term's increased use, papers relating to poorly worded agreements that are lacking information and have ambiguous meanings are not uncommon. For both the supplier and the client, this is a bad thing since it opens the door to negotiations that might damage the working relationship between them and, in the worst scenario, lead to the termination can be achieved through contractual agreements with external service providers or the decision to outsource to domestic service providers.

Users have less say over when and how services are delivered in the cloud, so they need to take steps to avoid issues like slow response times, inactivity, and data loss. As a result, the SLA must be considered a vital component of the cloud service delivery paradigm.

A service level agreement (SLA) establishes a contractual agreement between the service provider and the customer. A SLA is crucial for regulating the quality of services delivered since it tries to set the minimum acceptable for the proposed service. Each service has its own SLA that is created before the service is purchased and utilized. The service level agreement (SLA) should encompass the services provided, the assurances made by the providers guarantees, and the protocols and consequences to be followed in case of breach of these commitments.

SLOs (Service Level Objectives) are contained within SLAs and elaborate on the specific subjects that need to be observed and evaluated. The service provider is obligated to fulfill a Service Level Objective (SLO). The service architecture being offered has a significant impact on how SLOs are described. [22]

Considering the aforementioned concepts and comprehensive definition of service agreements, we can outline the stages involved in the life cycle of an SLA as follows.

1.5.1 Negotiate

The agreed-upon parameters for the service are laid forth in a service level agreement (SLA). Both the client and the service provider involved in the process must mutually agree upon the terms and conditions outlined in the service level agreement (SLA) that binds them together. Moreover , it is essential to clearly outline the assigned responsibilities and consequences in case of non-compliance with the specified rules and obligations. A lengthy procedure, this negotiation might take a considerable amount of time. An automated SLA's provision of an interface for parties to debate contentious SLA issues increases the likelihood that they will reach a mutually beneficial agreement fast.

1.5.2 Start

In accordance with the SLA, the service has been started and configured. If a service provider agrees to take on a contract, it must place the work in a queue and determine the delivery order for the services based on its scheduling policy. The provider must also think about maintaining the SLA-guaranteed quality of service while optimizing resource usage. It is vital to anticipate the arrival of new service request and prioritize them accordingly, even when other tasks are ongoing, even amidst ongoing tasks, to ensure the timely allocation of necessary resources.

1.5.3 Evaluate

Services are examined and analyzed as they are carried out to ensure that the SLA is being fulfilled. If a service provider has started giving its users access to new resources, it must ensure that those new resources are functioning properly. It is possible to utilize the data collected to check if the SLA's quality of service requirements is being met. Those concerned shouldn't care merely about whether or not a certain activity is being carried out appropriately. Verifying a SLA also requires information about potential breaches of contract or use data.

1.6. Approaches in SLA Composition

From a customer's perspective, the current SLA in commercial clouds like Amazon and Microsoft is straightforward because it is static and pre-defined by the providers. Following is a breakdown of the SLA's lifespan: The first thing a consumer should do is look for service providers that can meet their specific requirements. Customers use search engines to locate the provider, and once located, they make full use of the provider's website to learn all they can about them. Cloud companies often have an unchanging SLA agreement available for customers to peruse. As is customary when using external tools, the next phase entails choosing a monitoring and tracking provider.

By comparison to other commercial cloud providers, Amazon simply offers an availability SLA to its customers. Microsoft's Windows Azure takes response times into account, but only with some of its services. It's important to note that these SLA ideas are only texts and do not include dynamic measurements required for computing cloud services.[23]

Eucalyptus, a cloud computing middleware platform, offers a service level agreement (SLA) that is zone-specific. If a service-level agreement (SLA) is broken, it is feasible to move the affected services to a different availability zone, much like Amazon Elastic Cloud Compute (EC2) zones. However, the SLA does not have integrated monitoring, and neither does it automatically trigger service relocation. Similarly, Cloud Computing middleware monitoring simply reveals if a VM is in a pending, active, or off state. Getting data straight from hypervisors is essential for more precise monitoring. Data like as CPU use, memory usage, and network activity can be obtained from them.

It is clear that current cloud computing systems offer at least some SLA and quality of service, but they lack dynamic SLA negotiation. These SLAs have fixed-point solutions. Because of changes in demand, SLA breaches are more common in the cloud. Academics are working to standardize protocols for the automatic generation and tracking of SLAs for Web services. As a protocol for defining SLAs using Web Services and XML, WSLA (Web Service Level Agreement Language) generates an XML Schema that specifies the key components of the WSLA document include involve the parties, service assurances, and service description. The WSLA is typically structured into three main sections: Parties, Service Definition, and Obligations. Who provides the service is described under "Parties" (client or provider). The SLA's associated services are defined in the Service Definition, which represents the parties' mutual agreement on those services' essential characteristics. Last but not least, the Obligations section spells out the promised minimum standard of service with regard to the factors specified in the Service Definition. The WSLA provides for the administration of fines or compensation in the event of infractions. This, however, presupposes that the SLA already exists and does not permit the discussions to construct the SLA. Due to the unexpected nature of these changes, a static SLA designed to endure them will be ineffective.

The authors outline the WS-Agreement (Web Services Agreement Specification), an XML-based framework for the creation of SLAs and the SLA entails the commitments made by a web service provider to its customer regarding service guarantees. All the characteristics of a multi-service agreement, as well as references to earlier agreements, service descriptions, and warranty conditions, are laid out below. To make it easier to use, the specification is split into three parts: a framework for creating a standardized contract template and series of processes to manage the life cycle of agreement and a collection of states that may be generated, altered, and monitored.

These protocols often only deal with the rudimentary communication required to negotiate a basic service. A lot of work needs to be done before multi-service, multi-step auto-negotiation can be considered mature. Furthermore, neither protocol was designed with Cloud Computing in mind; it was developed for use only with Web Services.

When it comes to service level agreements (SLAs), the vast majority of commercial providers offer solely text-based agreements for cloud computing services and pass the responsibility of monitoring and regulating SLAs onto their clients. In this paper, we present SLAC, a novel language designed for composing Service Level Agreements in Cloud environments, and we compare it to some of the most well-known languages for this purpose, including: 1) SLA* is employed in versatile services, 2) SLAng is a domain specific language tailored for IT services, 3) WSLA is another notable approach, 4) WSOL is an XML notation that aligns with WSDL (Web Services Description Language), 5) WS-Agreement is yet another significant framework, 6) SLAC introduces a new language for composing Service Level Agreements in the cloud.

1.7 Frameworks of SLA in Cloud Computing

The frameworks addressed in this thesis were evaluated based on their ability to facilitate the recruiting process, the sophistication of the metrics or SLOs they use, the degree to which they are dynamic, the extent to which they integrate monitoring, and the number of metrics they support. In this approach, scientific ideas were selected based on their qualities that generally match the requirements but varied significantly according to the comparative study of the models.

As an example of a Cloud Computing framework that was disregarded, consider the CASViD framework, which is designed to track violations and report them. [24]

1.7.1 LOM2HIS

The LoM2HiS enables service level agreements to be hired based on quality of service (QoS) needs, also referred to high-level requirements, these focus on aspects such as reaction time, rather than getting into the details of low-level requirements, such as the number of processors or the size of the memory. The LoM2HiS architecture is seen in Figure 1.4

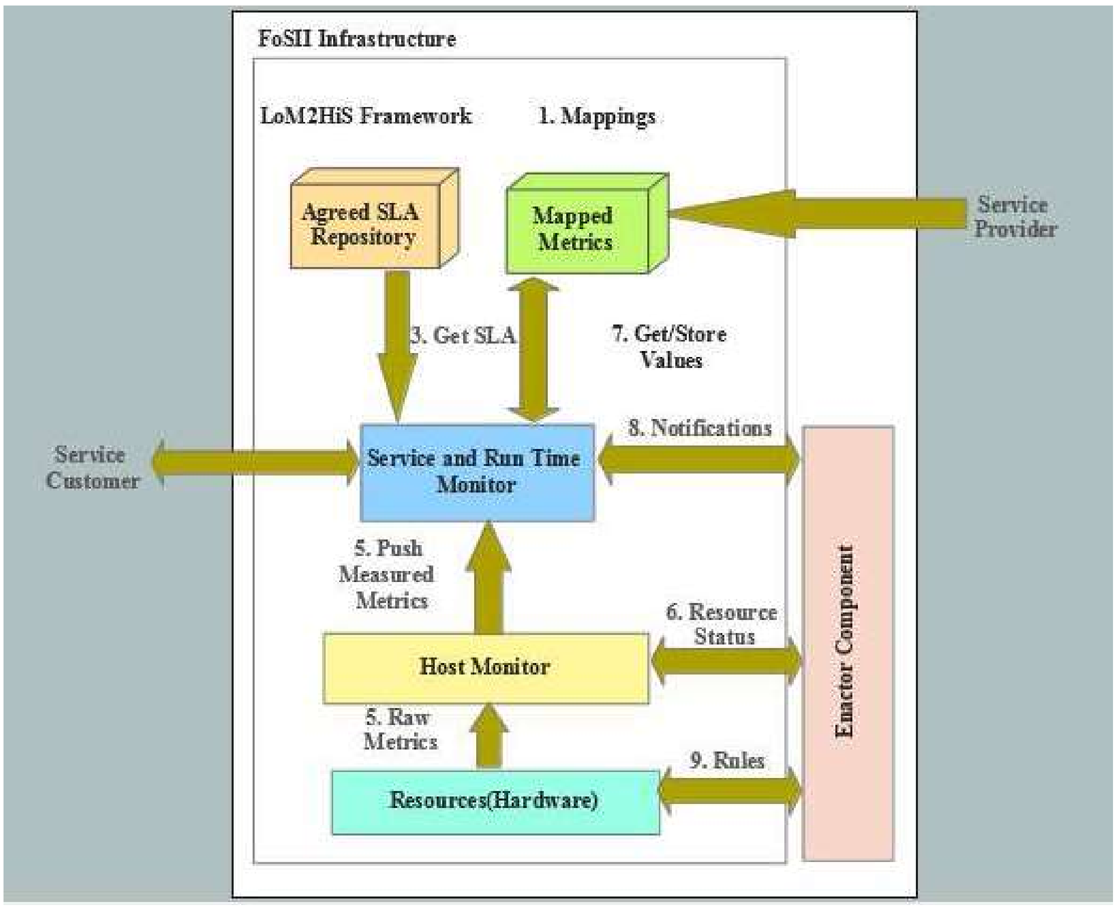


Figure 1.4: LoM2HiS Architecture

Using the methods described therein, we were able to accomplish:

Step 1: In accordance with the conditions of the Agreement, the service provider will develop domain-specific languages (DSLs) for the purpose of creating mapping rules for LoM2HiS.

Step 2: The client makes a request for the provision of a service as per the agreed terms;

Step 3: The runtime monitor populates the SLA repository with the agreed upon SLA;

Step 4: The availability of computers plays the crucial one in determining the ranges of services that can be provided. Monitoring agents continuously track the metrics associated with these assets;

Step 5: The host monitor retrieves the metrics and submit them to the run-time monitor for further processing;

Step 6: The host monitor provides updates on the resource's status to the enactor component;

Step 7: Both the low-level metrics monitor and the run-time monitor perform a comparison of the metrics using predefined rules to arrive at an identical service-level agreement. A mapping store is used to keep the result;

Step 8: The mapped values are used by the run-time monitor to track how the services are progressing while they are being executed. The enactor component is alerted to potential future SLA violations so that preventative measures can be taken;

Step 9: The enactor components execute decisions directly on the resources that currently accessible.

As the LoM2HiS assumes the completion of the negotiation process and the recording of agreed-upon SLAs in the service provisioning repository, it does not incorporate mechanisms for SLA negotiation.

1.7.2 DESVI

DesVi is an application that keeps an eye on your SLAs in the cloud to make sure they aren't being broken. The DeSVi will allocate resources and organize the service's deployment within a virtualized environment based on the user's request. After the baseline level of service has been established, indicators of potential Service Level Agreement (SLA) violations may be detected. Knowledge bases are used to handle service level agreement (SLA) violations, and these knowledge bases are built utilising learning techniques. DeSVI's capacity to identify and prevent SLA violations is shown by its execution outcomes in a varied environment. Operations including Service Level Agreement (SLA) trading, resource allocation, resource monitoring, and SLA breach detection make up DeSVi's service life cycle. In Figure 1.5, we see the structure and design of DeSVI.[25]

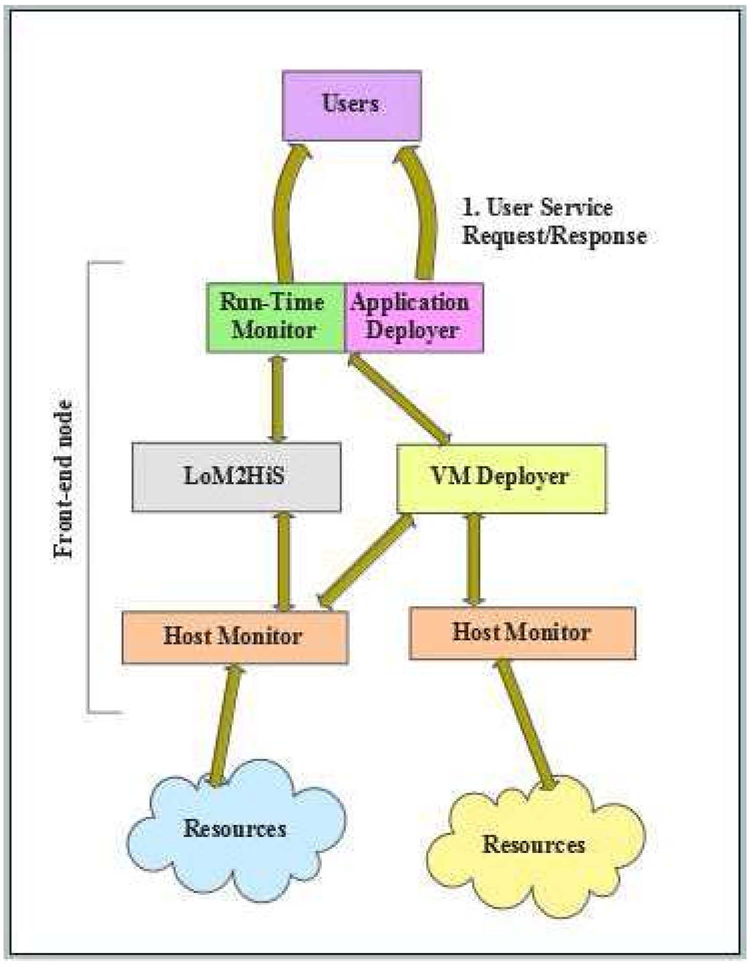


Figure 1.5: DeSVI architecture and interaction between components

At the top of Figure 1.5 are the users who make service requests to a service provider, who can respond in one of several possible ways.

Step 1: The user submits a service provisioning request to the cloud provider;

Step 2: The provider process user service requests based on the previously established and mutually agreed Service Level Agreements.(SLA) The program allocates the necessary assets as per the requirements and coordinates the deployment of the desired service on virtual machines;

Step 3: Subsequently, the deployment and configuration of virtual machines are carried out;

Step 4: Metrics for both virtual machines and physical hosts are recorded by the host monitor. The connection between resource measurements and service level agreements (SLAs) is managed by LoM2HiS.

The failover arrow in Figure 1.5 represents the built-in redundancy of the monitoring system. The metrics are monitored by the host monitor using monitoring agents that are built into the infrastructure. Such agents share the values they're monitoring with the remaining participants gain access to the current state of the same set of resources, allowing everyone to collaborate and utilize them.

DeSVI is able to monitor and identify SLA breaches with automated VM deployment. However, applications with high resource consumption variance are outside the scope of the DeSVI's analysis. Furthermore, the DeSVI does not deal with application-level monitoring or SLA violations (only infrastructure).

1.7.3. SLA-Based Resource Virtualization (SRV)

The SRV is made up of three primary parts: an automated deployment service that makes use of virtualization technology; a meta-trading component is employed to manage a universal SLA administration; and a diversified management component. These parts are depicted in an illustrative service architecture in Figure 1.6

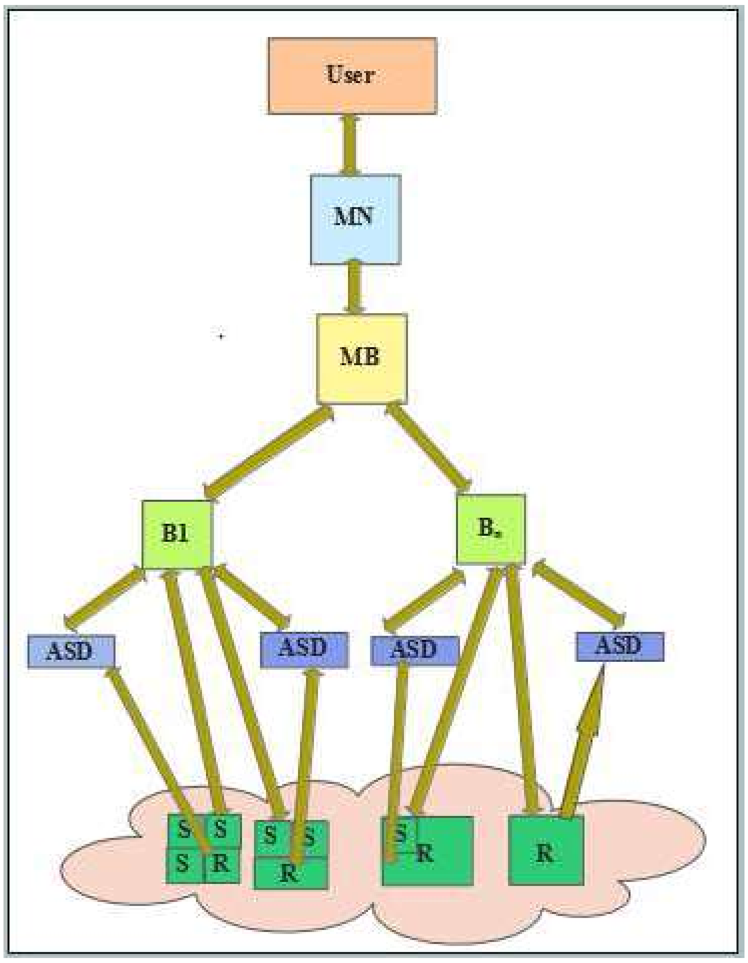


Figure 1.6: SRV Architecture

The "User" in Fig. 1.6 represents a potential customer for a service. The service level is controlled by the "Meta Negotiator" (MN). This component serves as an intermediary between user and Meta-broker, facilitating the selection of suitable protocols for agreements and engaging in negotiations for establishing the SLA and violation handling. a "Meta-Broker" (MB) is a part of the system whose job it is to find an agent that can deploy a service according to the user's specifications. When a service has to be deployed, it is "brokered" (B) through interaction with the Application Service Provider Device (ASPD). With Automatic Service Deployment (ASD), the needed service for the chosen component is automatically installed. Users refer to the service they wish to deploy and/or operate as "Service" (S). Physical computers that serve as hosts for virtual machines are referred to as "resources" (R).

This design illustrates the tight coupling between agreement negotiation and service deployment. While the SRV does provide a generic model for SLA negotiations in virtualized systems, it has not been examined in the context of Cloud Computing.

1.7.4. SLA for Scientific Research Clouds

Using prediction techniques that can predict whether or not a SLA will be tenable before the services are utilized, the authors present an SLA architecture designed specifically for SaaS scientific research clouds. The authors of this paper employ machine learning techniques to develop algorithms that govern the provisioning of virtual machines, hence maintaining the service level agreements (SLAs). Figure 1.7 , the SLA SaaS architecture is depicted ,specifically designed for a cloud-based research environment. [26]

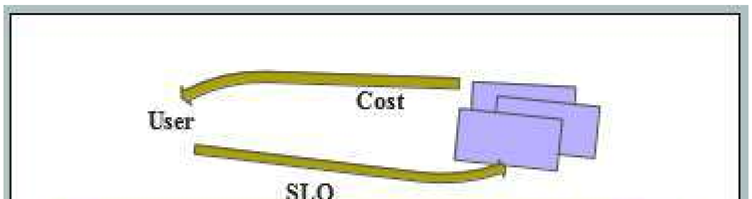
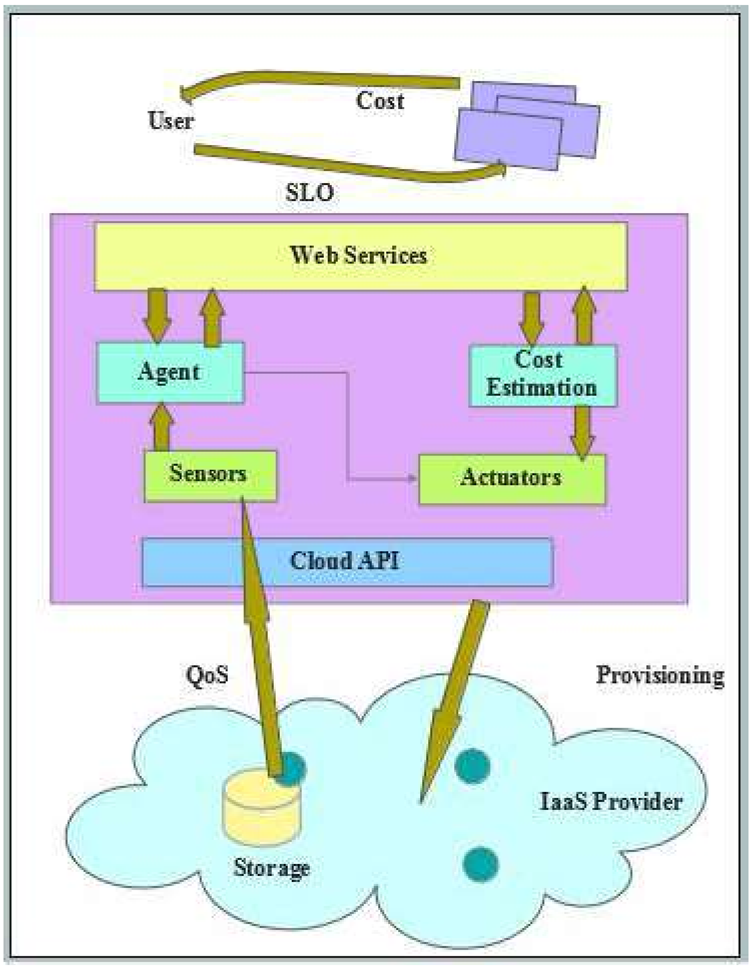



Figure 1.7: Research Clouds Architecture

Towards the bottom of Figure 1.7, an IaaS provider can be seen providing virtual machine instances within their own data centre, a key component of the cloud infrastructure. The infrastructure access is encapsulated in a layer just above, termed the cloud API. The cloud is where you'll find the applications' data storage. Sensors and actuators make up the subsequent layer. The parts communicate with one other and with instances already active in the cloud. In particular, sensors rely on the app since they regulate the app's status. Virtual machine instances can be created, modified, configured, and shut down with the help of actuators. In addition, there is an agent and a calculator for determining prices. During the active phase of service, the agent assumes responsibility for its management and ensures adherence to the service agreement. Initially, the cost evaluation module is utilized to access the feasibility of the user's requested action and provide approximate cost estimation. The agent and the cost estimation module may be accessed and communicated with using a Web Service interface built on top of the architecture. The design not only offers foreseen resource allocation, but also presents the user with predicted expenses. At this point, the task is initiated by the Agent module. The SLA is considered accepted once both the user and the service provider have clicked the "Submit" button. Agents are tasked with enforcing SLO adherence while also reducing resource use as much as possible. [27]

Scientific research SaaS providers and high-performance computing applications are the intended target audience for this architecture (High Performance Computing). As a result, work is allowed to continue uninterrupted until it is finished. As a result, the SLA is a one-and-done deal that lasts just till the project is over. The SLA cannot be renegotiated on the fly.

1.8. Comparison of Studied Frameworks

We highlight several features of the frameworks we've investigated in this part, including the following: the hiring interface, the skill level of the metrics, its inherent flexibility, its integrated monitoring, the cloud model it supports, and its strengths and shortcomings.

When discussing the process of acquiring a SLA, one often speaks of the "hiring interface." Technically terminology, including matrix like packets or received bytes, at lower level of granularity and transmitted, is commonly used to describe requirements or application metrics. It's also possible to use a variety of measuring units when assessing these indicators. Different units of measurement and presented in technical terminology might obstruct a fair comparison of how requirements of consumers and suppliers are discussed. In order to simplify comparisons and minimize the use of technical jargon, a high level requirement is established referred to as voice interface to Quality of Service (QoS). The LoM2HiS and the SLA provide a sophisticated recruiting interface for academic research clouds, respectively.

None of the models we looked at included dedicated KPIs for cloud computing in their evaluation of recruiting metrics. As an added bonus, the models were examined using the metrics independently of one another. As an illustration the qualifying system can assign varying weights, priorities, or levels of importance to specific groups or types of metrics. Subsequently, it can respond accordingly when in the event of breaches in certain metrics.

The model's adaptability to the elastic nature of the Cloud is seen in how easily the SLA can be negotiated and renegotiated. Out of all the models we looked at, only DESVI is built to deal with the whole SLA life cycle, from trading through resource allocation based on monitoring to violation predicting with machine learning.[28]

1.8.1 SLA Monitoring

The Service Level Agreement (SLA) should be reviewed at regular intervals to guarantee its continued validity. This calls for constant monitoring to verify the strategy is still relevant and workable. The monitoring data will be used to decide whether or not to continue the SLA or seek new terms. The ultimate objective of this function is to enhance the likelihood of fulfilling a service-level agreement (SLA) by optimizing the execution of a particular instance of an application, task, or service in alignment with the specific parameters. When a request is made to a service level agreement (SLA), it is evaluated so that the most suitable host may be chosen. The optimal configuration for a host is conditional on a variety of system characteristics, such as current availability, required resources for meeting SLA criteria, and desired optimization outcomes.

Reducing time to completion, decreasing expenses, and increasing the likelihood of success are all objectives that may be directly tied to familiarity with SLA standards. While other goals, such as workload balancing, are system-centric. To verify compliance with the terms of a SLA, it is become imperative to monitor the functioning of a service instance with respect to those defined terms. During execution, the SLA can be maintained or the effects of a breach mitigated by activating threat and recovery actions when a parameter associated with the service level objectives (SLOs) essentially the metrics to be monitored within the SLA-exceeds a predefined threshold values.

An intriguing method of monitoring is to inform the provider of potential danger so that he can take preventative measures. This feature also includes the option of telling the user who signed the SLA about the agreement's current state in order to increase openness and trust.

Exemplifying these characteristics we report the results of a survey on Cloud monitoring, in which the authors compared and contrasted proprietary and free monitoring software. The authors discovered criteria that must be provided by monitoring systems in order to do these analyses. The following are examples of such characteristics:

1.8.1.1 Adaptability:

According to the authors, a flexible monitoring system can adjust to different amounts of data being processed and transmitted at any one time.

1.8.1.2 Autonomicity:

The distributed resources of an autonomic monitoring system are managed autonomously in response to sudden and unexpected events.

1.8.1.3 Comprehensiveness:

Multi-tenant support, multiple resource types, and multiple types of monitoring data are all indicators of a system's comprehensiveness;

1.8.1.4 Resilience:

A system of monitoring is considered robust if it can be reasonably relied upon to maintain service delivery in the face of alteration;

1.8.1.5 Reliability:

When a monitoring system consistently provides the expected output over time and under the specified operating conditions, we say that it is reliable.

1.8.1.6 Availability:

Since monitoring services are always at hand, users can always get what they need when they want it, as long as it fits the system's requirements.

1.8.1.7 Accuracy:

When a monitoring system's measurements are within a predetermined tolerance of the true value being measured, we say that the system is accurate.

1.9. Measures and Preventive Methods in Cloud Security

1.9.1 Cloud security measures

The term "cloud security" refers to the technology and policy that regulate and limit user access to different types of internet services in a secure and trustworthy way. When compared to on-premises systems, public cloud computing has a higher attack surface area and worse security. Common sources of cloud security issues include inadequate security measures and careless or unskilled users inside the organization that relies on or plans to embrace secure cloud computing.

1.9.1.1 Cyber Security in Public

Each company launching its cloud-based services to the general public must take this complexity into account as a prerequisite for doing so. Cloud computing can be performed in various environments; including public cloud, a private cloud, or a hybrid cloud. Public cloud denotes a specific type of cloud computing where services are provided to is user through an existing network infrastructure, typically over the internet.

The concept of a "private cloud" refers to a distinct form of cloud computing architecture whose primary purpose is to serve the needs of a single company over an internal wireless network. As a cloud-based strategy, a hybrid cloud integrates elements from both public and private cloud environments to better serve its users. Most businesses run 38% of their operations in the public cloud and 41% in private clouds, with mission-critical tasks performed in the former and less important tasks in the latter.

1.9.1.2 Cyber Security Concerns

About 70% of all businesses use cloud computing technologies because of their adaptability, compliance, and interoperability, making it crucial to comprehend the many cloud security considerations for secure cloud computing practices.

1.9.1.3 The benefit of Cloud-Based Secure System

A Cloud-Based Secure system's main advantage is that it can speed up the deployment of services to customers, cut down on the resources expended on maintenance, and improve the efficiency with which new software is introduced. Besides the obvious security advantages, a cloud-based system also offers users more insight into how their systems are being used. To fulfill cloud compliance requirements, businesses can either permanently store all of their data and workloads in the cloud, or move them from one cloud system to another.

1.9.1.4 Cloud Provider Criteria & Preferences

The seven most commonly utilized cloud providers include Amazon AWS, Microsoft Azure, Google Cloud, Oracle Cloud, IBM Cloud, Rackspace, and Alibaba . According to the survey, 63% of participants indicated that cost was the primary factor influencing their choice of cloud provider. This was followed by ease of deployments (53%), availability of security tools and cloud-native features (52%), and automated deployments (52%).Other consideration included interoperability with on-premises system (48%), customization of policies(44%), integration with cloud platforms (44%), and support for multi-cloud(42%). Companies have prioritized attacks on Amazon Web Services above those on other cloud service providers, as reported by a poll of 653 cyber security professionals conducted by Cyber Security Insiders.

1.9.1.5 Cloud Solution Criteria Based on Native vs Independent Cloud Security Solutions

Cost, ease of use, user friendliness, and well-integrated software that is designed for simplicity of operation are the most crucial factors for selecting a Native and Independent Cloud Security Solution. Other cloud security solutions focus on things like speed of deployment, the safety of cloud vendors, and the dependability of cloud services. The term "Native Cloud Security" is used to describe a unified cloud service that provides protection for cloud-based applications in real time. Each module relies on the others to function properly; the native cloud security solution is tailored to micro service architectures, while independent clouds are designed for narrow use cases and cloud-enabled applications are developed on top of preexisting, on-premises backend systems.

2.0 Cloud security preventive methods

In order to use cloud computing services securely, businesses must be aware of the many threats they face and the methods available to protect themselves from those threats. A crucial requirement for mitigating potential hazards in cloud computing is the establishment of legally binding agreement involving the appropriate parties to prevent any potential liabilities in the future. Limiting user access, investigating system vulnerabilities, analyzing the open port, and creating an incident response plan are all common practices for reducing risk in cloud computing.

2.1 Strengthen Access Control

To establish the secure cloud computing system, it is imperative for security focused stakeholders within a company to perform comprehensive evaluation of the current access control system. Subsequently they must develop a robust set of controls that effectively address any weak configurations, thereby enhancing the overall security posture of the system. Multifactor authentication, stricter password regulations, encrypted password storage, user access restrictions, and account lockout policies are all on the rise as methods for bolstering security. Below, in Figure 1.8, we demonstrate the most prevalent tendencies of bolstering access control.

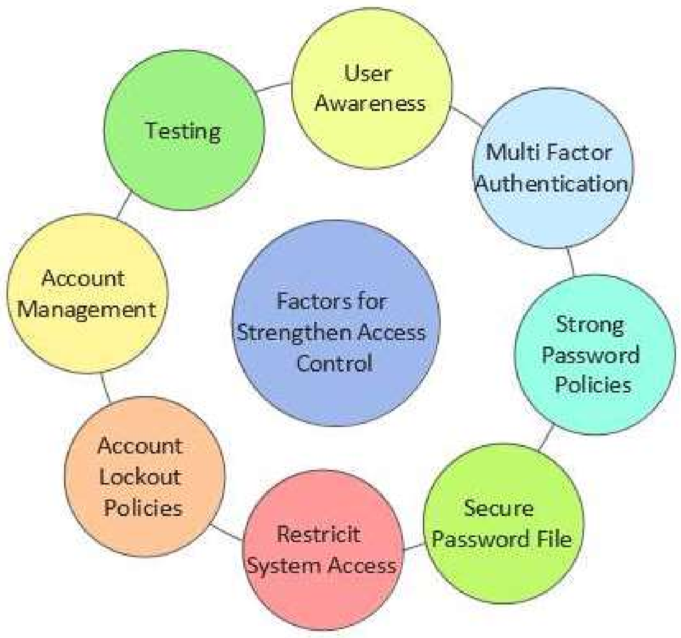


Figure 1.8: Factors for Strengthening Access Control

2.2 Multi-Factor Authentication

By requiring authentication from many sources, or "factors," before giving access to a resource, Multi-Factor Authentication (MFA) increases security. An alternative to the more prevalent and conventional way of authentication including a username and password is to employ a combination of account addresses to determine if the user is genuine. These account addresses may include Gmail, Mobile Number, and other open-source connections.

2.3 Stronger Password

Utilizing an automation script to accurately determine the appropriate characters for password fields is one more effective method of finding off a variety of cloud computing attacks, such as a brute force. In order to prevent attacks on the cloud infrastructure, such as those based on brute force, most sensitive accounts only allow multiple attempts at login, usually no more than three.

2.4 Active Surveillance

Monitor network traffic, encompassing both inbound and outbound data packets, and analyzing vulnerable ports are to safeguard the cloud computing system against potential data leakage, often unbeknownst to the users. Monitoring idle connection ports and doing a thorough analysis of each data packet is crucial for the security of cloud-based data.

2.5 Data Integrity

Information stored in the cloud must be encrypted using a sophisticated set of algorithms to prevent malicious software from accessing the data and to increase the security of the files in the event that they fall into the wrong hands. The majority of businesses using cloud computing also use a massive amount of data and information from its customers and internal employees, making encryption of data essential.

2.6 Data Encryption

An organization that relies on cloud computing services to store and manage its data must be familiar with data encryption in order to protect the data and maintain its information integrity. Even if confidential information falls into the wrong hands, it may be protected through the use of strong encryption techniques. Even more, the most well-known cloud-based companies prioritize data security and implement sophisticated encryption techniques as a part of their comprehensive security measures that make decrypting the encryption key nearly impossible.

2.1 Cloud Orchestration

In the twenty years since the cloud-computing paradigm evolved, cloud infrastructure installations have become larger and more complex. There has been a dramatic rise in complexity as a result of the proliferation of interconnected components, the breadth of APIs available, and the number of new mechanisms for accommodating variable scaling and optimizing computational performance. This meant that cloud orchestration was required across the whole cloud infrastructure lifetime, not just the deployment phase.

Over time, administrators in the cloud transitioned from manually deploying, configuring, and monitoring cloud instrumentation components to setting up orchestrator frameworks and deployment templates. One way to do this is to write configuration policies that instruct the orchestrator on what to do in order to realise the cloud infrastructure, while another is to express high-level imperative intents that describe the desired state of the infrastructure while leaving the implementation details to the orchestrator. Today's orchestrator solutions cover the whole lifecycle of cloud infrastructure, from deployment through monitoring and load balancing to the continuous deployment of new workloads. Automated orchestrators let you keep your cloud services running smoothly and reliably with little to no human intervention, all according to your configuration policies or intentions.

While orchestration is used by cloud operators to build up and run cloud infrastructure, it may also be used by attackers who compromise the system by using incorrect or maliciously modified orchestrators or by forging configuration policies and intentions. By exploiting a flaw in the orchestrator, hackers can compromise a system.[30]

2.1.1 Orchestration Models

Micro service-based, dynamic applications are replacing monolithic legacy program that were transferred to cloud platforms, and this is where cloud orchestration comes in. Older program had rigid layering and invoke between layers using layer-specific APIs. Similarly, service characteristics, functionalities, and physical resources are all owned by the same organization.

However, dynamic applications are constructed using micro services that only offer rudimentary functionality. Instead of employing layers, they rely on recursion, and functionality is accessed via generic service interfaces that are shared between different levels of the stack. In addition, features are separated from resources, thus the people responsible for the features and functioning may not be the same people who control the underlying hardware. At each level of decomposition, the orchestrator can choose whether to take over management of the component, or pass it off to another orchestrator in a different domain, which may then utilize recursive decomposition to deploy the component on its own resources.[31]

In light of the aforementioned, we may classify orchestrators into two broad categories: imperative orchestrators and declarative orchestrators. Imperative orchestrators pay special attention to the deployment method, requiring specific instructions on how to deploy services and what the orchestrator should do at each step of the process. This strategy is shown by the rise of "infrastructure as code" and other forms of automated system administration. Declarative orchestrators are those that place emphasis on the target infrastructure and envisioning the intended outcome of the planned service deployment, leaving the interpretation and execution to the orchestration.[32]

2.1.2. Security Enablers in Multi Cloud Orchestrators

Next, we present a collection of security enablers for establishing trust in multi cloud orchestrators.

2.1.2.1 Crypto Engine

The Crypto Engine is an independent micro service that offers the following functionality by implementing a set of cryptographic algorithms:

- Performing calculations of cryptographic hash functions
- Generating the Message Authentication Code (MACs) for message integrity
- Generating cryptographic key
- Generating nonce for cryptographic operations

2.1.2.2 Credential Manager

Credentials for entities with access to the cloud orchestration service are kept in the Credential Manager (CM). The CM is open to requests from any organization and is accountable for the secure and confidential rollout of the associated credential. The CM also maintains all credentials in a manner that enables it to solely confirm their validity without exposing any of the details of the credential.

2.1.2.3 Firewall Service

This service guarantees that the firewall configurations of virtual containers, created by the orchestrator, remain constantly updated in alignment with the most recent security policies defined by user intentions or topology specifications.

2.1.2.4 PKI Manager

Using a standard challenge response protocol between the requesting party and the issuing party, the PKI Manager can issue or revoke credentials as needed.

2.1.2.5 Attestation Service

Even though hypervisors (or operating systems) help keep things separate in a virtual execution environment, that alone isn't enough to build trust between a tenant and the computing resource in question. To close this security hole in cloud orchestration, we leverage remote attestation. An appraiser assesses one or more targets within the cloud, whereby an entity (typically a computer or network) evaluates and makes determinations regarding the target or targets. A target such as computer system necessitates an evaluator make a critical assessment or judgment. An attestation protocol is a means via which an appraiser and a target can exchange information.

2.1.2.6 Image Integrity Verifier

Cloud infrastructures are especially vulnerable to the dangers posed by corrupted picture files. The orchestrator can check the integrity of the image files used to create virtual containers with the help of the image integrity feature. Virtual machines, lightweight virtualization containers, and unikernels all fall under our umbrella term "virtual containers," but stateless functions that rely only on process separation are left out.

2.1.2.7 Image Delta Verifier

In this way, the attestation service can determine which images have been modified in some way. Validating variations between virtual machine images or lightweight virtualization containers is one concrete example of this.[33]

2.2 Orchestration Security Architecture

Next, we take a look at the suggested security architecture and detail how its primary parts interact with one another (see Figure 1.9). For future orchestrator frameworks, we propose security architecture. It can be modified to newer cloud orchestrators, but it can also be used with older ones. It's also worth noting that this study is narrowly focused on security-related issues. If you want a more detailed explanation of what makes up an orchestrator framework, consider the following:

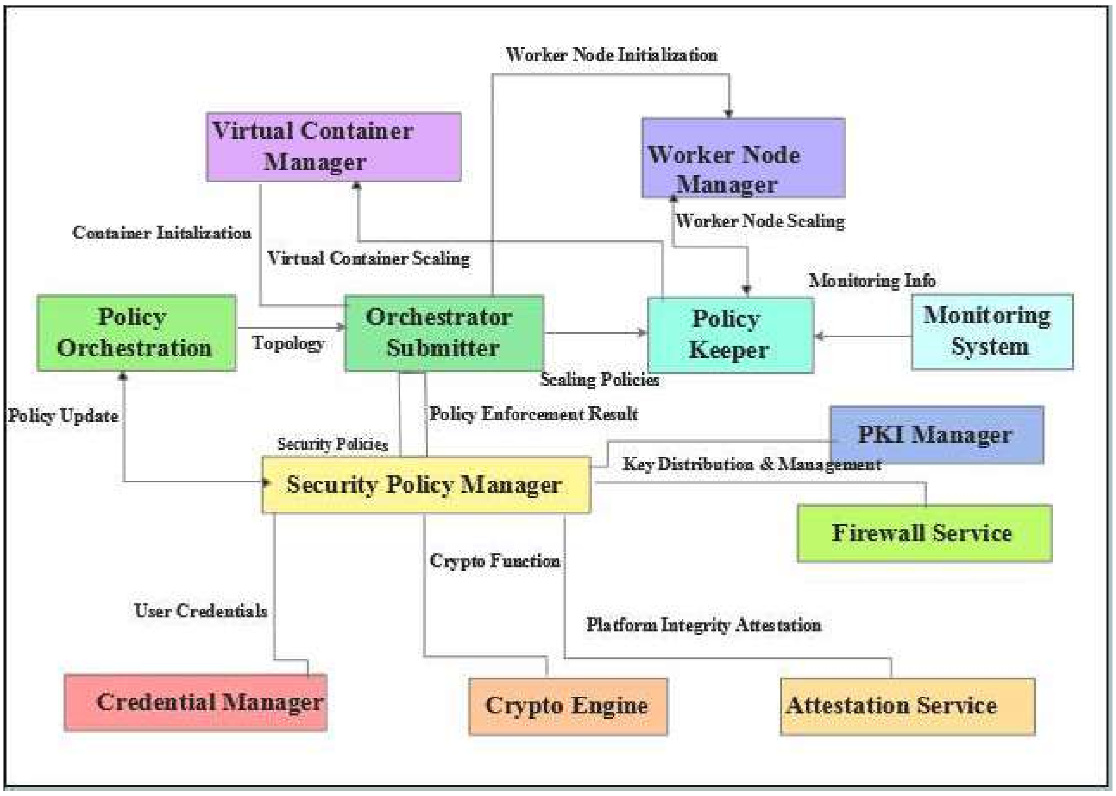


Figure 1.9: Architecture of Security

2.2.1 Secure Component Deployment

Our goal is to provide a means for off-site system administrators to trustfully launch cloud-based software and to obtain assurances regarding the safety of the underlying network and operating system. The platform's master nodes are assumed to be up and running at this point. The PKI manager distributes the public keys of the architecture parts during the deployment phase of the infrastructure. Administrators set security policies for key length, algorithm selection, and entropy requirements, and the Crypto Engine generates public keys in accordance with those guidelines. To specify orchestration goals, system administrators write policies in a domain-specific language like TOPOS-CA (Topology and Orchestration Specification for Cloud Applications).

Users must first create a template that details the application's topology and security policies before the application can be deployed. The public key of the orchestrator submitter is then used to encrypt the template file before it is transferred across a secure communication channel to the orchestrator submitter. Users can send application data and worker node VM configurations to an orchestrator, which, once received, will be parsed for any embedded security regulations. The Security Policy Manager is responsible for defining and enforcing security policies, while the Orchestrator Submitter is responsible for parse the User Template and provide the appropriate security policy data and access credentials to it.

The Security Policy Manager ensures the integrity of the transmitted security policies immediately after receiving the message. If the policy checks out, the Security Policy Manager will utilize the credentials to access the application and other information (such the image identification of the virtual container) to put the user's security rules into effect. The Firewall Service configures these parameters and controls access to the network.[34]

2.2.2 Component Interaction

Here are some examples of how the Security Policy Manager interacts with orchestration components and calls upon security enablers:

The Security Policy Manager checks the virtual container for correct deployment before any applications are deployed. Therefore, before launching a virtual container, the Security Policy Manager uses a feature of the Attestation Service, to ensure that the image of the container has not been tampered with in any way. Once the virtual container has been successfully validated, the Security Policy Manager proceeds to authenticate the user with the cloud service provider and initiate the creation of virtual container using the specified image. To create or remove virtual machines (VMs) on remote resources, the VM orchestrator can use the user's cloud credentials to verify their identity with the supported cloud service providers. Simultaneously, the virtual container manager dynamically manages the creation, removal, or migration of virtual containers based on the resources utilization of the active application. The attestation authority can be used by the system administrator before the application is launched to verify the host's software is authentic. By doing so, the user is assured of the integrity of the entire host system.

2.3 Objectives of the Study-

Following research objectives would facilitate the development of this aim.

1. Identifying factors impacting on cloud computing SLA.
2. Identifying factors impacting on security orchestration.
3. Analyze the protocol in security orchestration.
4. Analyze the protocol in cloud computing SLA.
5. Develop the protocol to enhanced security orchestration in cloud computing SLA.

2.4 Research Methodology-

This thesis created three intrinsically valuable strategies for automating SLA management as its research objectives. First and foremost the contributions of this work are characterized by their generality enabling wide ranging applicability across various domains without being restricted to any specific field. The negotiation method developed in this project was successfully integrated into the Contrail project following its validation across numerous SLA@SOI use cases. Second, the engineering and algorithmic contributions will offer several granularity levels of policy-based controls. This is required to adjust the procedures that have been designed in accordance with business-specific instructions, notably for customizing negotiation sessions, connecting utilities to contract spaces, or performing multiple-criteria infrastructure operation optimizations.

2.4.1 Proposed Techniques

A general technique that takes negotiation protocols from conception through execution as verifiable, adaptable, and machine-executable entities. By creating a bilateral negotiation procedure, this will be shown. Additionally, a platform for negotiations will be created to enable point-to-point negotiation (s). The platform will be utilized in distributed SLA scenarios with dependence on several providers for chained or layered negotiations. The protocol, platform, and protocol development methods are all described. The first research question is therefore answered.

There will be two negotiation strategy algorithms created. These do this by quickly moving across contract areas and adjusting to the concession-making behavior of the negotiation party to optimize the commercial value of a SLA. The second research question will be answered by the techniques and their assessments, which will be provided.

Platform as a Service (PaaS) cloud infrastructures will be the primary focus of a resource management system that is cognizant of SLAs. Four Meta-heuristic search methods will be used as part of the solution to plan and optimize resource reallocation in a recurrent "service consolidation" scenario. Regarding installed services and cloud machines, a range of soft and hard limits are considered. Simulations will be used to assess performance in relation to a range of quality criteria, and solutions will be graded in accordance with high level rules. The third research question will be answered by the findings that are given.[35]

Two scenarios will be presented in this essay. First a system model in which two entities may carry out their service exchange under specific commercial terms and operational circumstances while utilizing a variety of tools, methodologies, and processes (ii) Next, it will talk about fault models for situations when the same actors encounter specific (internal or external) opponents and examine their (machine, human) behavior to identify potential dangers. The protocol includes a trusted third party (TTP) responsible overseeing the entire transaction and ensuring the fairness component is upheld until the transaction is successfully concluded.

Finally, this study will explore how this procedure variant protects against situations when one or both participants decide to act deliberately rather than being loss-averse in order to obtain an unfair advantage over the other. The protocol will also show off its defenses against other critical matters like security and privacy to guarantee its dependability and transparency while implementing cloud SLAs.

CHAPTER -2

2.0 Literature Review

Al-hamideh and E.W.G.(2022)The approach taken to assessing IaaS in the cloud was straightforward, time-consuming, and sequential. Tools included Iperf3, Netperf, MTR, WinSCP, and Putty; locations ranged from Canada to Tokyo; and there were three distinct server configurations tested: "general purpose," "memory optimize," and "compute optimize" (10,50,100). On par with, or parallel to, both of the competing service suppliers.

Ravele et al. (2022)As a result of cloud computing, we now store all of our systems and data on remote servers located all over the world. The study set out to discover what elements are most indicative of successful cloud computing on the part of individual users, and how this relates to the overall performance of the platform. The cross-sectional, descriptive, quantitative survey was carried out in communities across South Africa, and 254 responses were collected (with a response rate of 66.1%). The findings of the measurement model (SRMR = 0.071) showed that the quality of cloud computing platforms used by individual customers is best defined as a combination of low cost, dependability, adaptability, availability, security, and scalability. Convergent validity, reliability, and discriminate validity were all validated by AVE (0.575-0.694), CR (0.844-0.901), Fornell-Larcker criteria, and cross-loadings. Statistics show that there is a modest to medium impact size between how well the platform secures data and how well it scales, how reliable it is, and how fast it processes requests. An improved cloud-computing-based domestication theory of technology has theoretical implications for understanding what's needed to adapt to existing environments and contexts. The data also shed light on how individuals were making cloud computing decisions. Ultimately, the research improves cloud computing for individuals by helping them realize that their own cloud computing decisions and values may differ from those of established businesses.

Dadhich et al.(2021) The primary goal of this article is to identify the most influential technological, organizational, environmental, and economical components on library cloud computing (LCC) among library patrons and professionals at a set of Indian colleges. In this study, we explore the merits, prospects, and threats, as well as the models, of a Smart Library in the information and communications technology era. To determine the strength of the links between the study's chosen components and LCC uptake, the researchers used exploratory factor analysis (EFA), canonical correlation (CC), and structural equation modeling (SEM). By focusing on the technical, organizational, environmental, and financial structures, as well as the 16 manifestations in the given model, empirical study has presented four theories. After developing the model, SEM-ANN was used to put it to the test using data from 510 students at 26 public, private, and state institutions across India. The first step was to use SEM to identify the contributing factors of LCC. Second, the ANN output ranked the influential predictors found by SEM. The results also show that users and stakeholders can gain fresh insights from people's behavioral intentions while using a library's cloud services.

Sayginer, C.,& Ercan,T.(2020) Using the Diffusion of Innovation (DOI) and Theory of the Firm (TOF) models, this study will examine the internal and external factors that influence CC adoption decisions among businesses in Izmir, Turkey. This confirmatory study polled 176 IT decision-makers from companies in Izmir, Turkey, that either don't utilize the cloud or are already cloud users. The factors influencing the shift to cloud computing are analyzed with the help of SmartPLS 3.0. Summary of Major Results: Security and privacy worries, as well as cost savings, were shown to act as mediators of relative advantage for CC adoption, alongside the study's other identified factors of relative advantage—compatibility, complexity, and support from upper management. According to the results of the poll, both complexity and the backing of upper management are crucial for successful CC implementation. The model accounted for 41.2% of the variance in CC uptake. The significance of these findings: It is anticipated that this research will be of help to those in company administration, sales, marketing, and IT infrastructure, as well as those developing businesses across a variety of industries. In particular, our research and the proposed approach will assist firms in making sensible CC adoption decisions. As the research also suggests, governments might make use of such frameworks to persuade cloud service providers to aid businesses during the decision-making and transition phases associated with CC adoption.

Wan Mohd Isa, et al (2020) Few research have examined what factors affect the uptake of cloud computing in academic settings. However, there is a lack of awareness of the challenges associated with cloud computing adoption on campus. The primary goal of this research was to identify barriers to cloud computing adoption in a university setting. The study included a case study approach and qualitative interviews with key participants at a single public institution in Malaysia. The Atlat.ti software was used for the analysis. There are nineteen different elements, all of which have been assigned codes that fall under one of three broad headings: technological, organizational, and environmental. When deciding whether or not to utilize cloud computing, these are some of the considerations a public institution will make. The findings might serve as a benchmark for expanding cloud computing use in mobile education and computing. In the future, we hope to replicate this research at additional Malaysian institutions of higher education, both public and private.

Islam, Chadni, et al. (2020) Security Operation Centers (SOCs) employ a wide range of techniques to detect, prevent, and respond to security attacks. Faster integration of security technology and operational duties is a big challenge for the SOC. Security Orchestration, Automation, and Response (SOAR) systems are being used by more and more enterprises to address this problem, but their design needs solid architectural backing. Our research on the advantages of architecture-centric support throughout the SOAR platform design process is presented in this article. Our method includes a conceptual model of the SOAR platform as well as a list of the major architectural design space criteria. We have shown the effectiveness of this method by building and releasing a Proof of Concept (PoC) SOAR platform for I automated security tool integration and (ii) automated activity interpretation to carry out incident response activities. We also provide a preliminary assessment of the suggested architectural aid for enhancing a SOAR's design.

Chadni Islam (2020) Companies use a wide range of security measures to thwart cybercriminals. Different companies' security products use a wide range of technology and conceptual approaches. Thus, achieving seamless operation between different security measures is challenging at best. Security orchestration aims to help the security staff of a Security Operation Center by integrating multivendor security technologies that are able to interoperate rapidly and effectively (SOC). There has been an increase in the amount of published content on various security orchestration systems due to the significance and relevance of security orchestration. However, little work has been put into thoroughly examining and assessing the purported answers. In this work, we provide the results of a multi vocal literature analysis that began in January 2007 and went through July 2017; it aimed to identify and analyze published academic and grey (blogs, websites, white papers) literature on a broad variety of security orchestration subjects. Based on these findings, we are able to define security orchestration and separate its core components into the categories of automation, orchestration, and unification. We have also broken down the factors that influence security orchestration into technical and socio-technical categories, as well as defined the components of a security orchestration platform. We also give a security orchestration taxonomy that classifies solutions according to execution setting, automation method, deployment type, task mode, and resource type. We were able to pinpoint several opportunities for further study and advancement in the subject of security orchestration by conducting this evaluation.

Hassan et al., (2019) The proliferation of cloud computing has forced a reappraisal of software distribution, development methods, and supporting infrastructure. Solutions that are adaptable, easy to manage, robust, and reasonably priced are all within reach, thanks to cloud computing. The rapid spread of cloud computing has prompted concerns about data safety. In a shared cloud environment with many other users, there are more opportunities for data and application theft. Several studies have been undertaken to evaluate the security challenges that cloud computing systems face, and a plethora of proposed solutions have also been made public. Increases in security threats need the creation of comprehensive security measures. This page discusses cloud computing from every angle, from the fundamental concepts to the most useful services. This research not only assesses the many threats to cloud computing security, but it also examines potential countermeasures. Also included is a brief summary of the relevant regulatory agencies and compliance requirements for cloud security. This presentation covers various unresolved research difficulties and barriers in the field of cloud security.

Skafi, M., et al (2019) The adoption of cloud computing is discussed, and a survey of the relevant literature is offered. The growing popularity of cloud computing amongst enterprises worldwide, and in the MENA region in particular, highlights the significance of investigating the variables that are likely to affect the adoption of this computing paradigm amongst SMEs operating in a variety of industries and sectors. To achieve cost effectiveness and higher returns on investments in information technologies and systems, researchers largely agree that the Software as a Service (SaaS) model of cloud computing has a considerable influence. This computing model is still relatively new and contributes for a tiny fraction of overall IT spending, despite its rapid development compared to traditional computing models. The enablers and restraints influencing the option to adopt cloud computing are still an issue that needs to be addressed in the MENA region, especially in the SMEs segment of organizations. This literature synthesis does this by defining the facilitating and impeding factors that impact cloud computing adoption and by highlighting the knowledge gap about factors that may be especially relevant to Lebanon, a Middle Eastern country. The acceptance of cloud computing, investment in cloud computing services, enablers, difficulties, and cloud computing adoption in the MENA area were the main points of focus in a literature study of a few carefully chosen articles. Google Scholar was used as the first stop in the evaluation process, followed by other available research databases including EBSCOHOST, Emerald, and the ACM Digital Library. This page provides a bibliography of the most influential books published on these subjects, mostly between the years 2011 and 2017. We cover 22 papers here. The primary objective of this article is to provide academics in Lebanon and the MENA area with a framework for examining the factors that influence small and medium-sized enterprise (SME) adoption of the cloud computing paradigm. This study also outlines the primary theoretical frameworks used in this line of inquiry and the important factors that explain the dynamics that are driving or limiting adoption. By classifying reviews according to aspects emphasised and the theoretical framework employed by the authors, this study develops a model based on the synthesis done of the various components acquired from the literature and the relevant theoretical frameworks.

Islam et al., (2019) Businesses use several different types of security measures to prevent cyber attacks. The many different types of security solutions available today are the result of a wide range of technological and conceptual foundations. That's why it's so challenging, if not impossible, to achieve seamless operation across different security measures. Due to the importance and relevance of security orchestration, the amount of written material on different security orchestration systems has increased. However, there has been little effort to properly examine and assess the alleged solutions. This research allowed us to define security orchestration and classify its core capabilities as either automated, orchestrated, or unified. We have also classified the variables that impact security orchestration into technical and socio-technical groups, and identified the core features of a security orchestration platform.

Senarathna et al.,(2018) Cloud computing is a newer computing paradigm that provides high-end computer capabilities to organizations through the Internet for a pay-per-service cost. It allows SMEs to catch up to their larger competitors in terms of technological innovation without making huge financial commitments. In this analysis, we analyzed the critical factors that influence the rate at which SMEs embrace cloud computing. The data was analyzed using a multiple regression model, and the results revealed that small and medium-sized enterprises were affected less by risk factors than by factors relating to improving their organizational competency. The findings might be useful for small and medium business owners, cloud service providers, and policymakers as they work to help SMEs migrate to the cloud.

Mubeen, et al., (2017) Technology advancements in the realm of computing, such as cloud computing and the Internet of Things (IoT), have enabled businesses to better respond to consumer demand and market shifts by providing a wider range of services to their customers. Therefore, it is essential for service providers and service consumers to establish QoS via Service Level Agreements (SLAs) for such cloud-based services. Since SLAs are essential for cloud deployments and increased usage of cloud services, their management in cloud and IoT has become an important and fundamental component. SLA management for cloud-based IoT applications is explored in this article. Systematic Mapping studies are used by academics as a regular process for sorting through the literature in search of studies that are relevant to SLAs. Major research on service level agreements (SLAs) is broken down into seven technical areas (SLA management, SLA definition, SLA modeling, SLA negotiation, SLA monitoring, SLA violation and trustworthiness, and SLA evolution), with a total of 328 papers listed. Research methods, key results, and demographics of study participants are also summarized in the report. The review of the data reveals that most approaches to managing SLAs are used in academic or controlled research with constrained industry settings rather than in actual industrial settings. There has been a lot written on SLA management, but practitioners still have a hard time since the evolution perspective hasn't been well explored and there isn't enough tool support. Very few studies have also concentrated on genuine metrics for qualitative or quantitative evaluation of QoS in SLAs, hence there is an urgent need for research into the development and measurements of metrics for SLAs.

Kanagasabapathy et al., (2016) In cloud computing, you're supposed to pool your resources for maximum efficiency. Being adaptable, cheap, and quick to store data, it has quickly become one of the most popular services on the web. The widespread use of third-party cloud service providers raises security concerns. Since they go over the same traditional network protocols as cloud data, if those protocols are hacked, cloud data is equally at danger. This research looks at the major security issues with cloud computing and offers solutions to them.

RadhyaSahal et al., (2016) A new form of computing called cloud computing uses a layered paradigm and offers its users a variety of services. The idea behind cloud computing is to provide a scalable platform for Big Data Analytic Applications (BDAAs) that may elastically provide resources dependent on the complexity of the analytic applications and the expansion of the data. Large data quantities and huge businesses' preference for quick, efficient decision-making cause the complexity of analysis to rise. In order to clarify the duties between a client (such as a cloud user or a big data analyst) and a provider for a certain service supply, the relevance of Service Level Agreements (SLAs) has emerged. A set of Service Level Objectives (SLOs) must be met, and Quality of Service metrics must be monitored to find breaches. In fact, a variety of SLA management techniques have been created as remedies for preventing SLA infractions to avoid exorbitant fines. As a result, several creative ways for managing SLA violations in cloud technology and cloud-hosted BDAAs were created. In this study, an overview of the ideas, benefits, and drawbacks of the existing works is presented. To give a thorough overview and a big-picture perspective, the obstacles and fresh research paths in this field that need more examination will be presented in the meanwhile.

Conte de Leon et al., (2016) Today, sensitive data and systems, such as intranets and crucial control systems, are accessed and modified via web browsers. Browsers are susceptible to a variety of assaults because of their processing power and network connection, even when they are properly patched. Phishing attacks mainly target browsers as well. By utilizing site-, user-, and device-specific security measures in a varied browsing ecology, many browser assaults, including phishing, might be avoided or lessened. However, our study showed that the security configuration processes, option names, values, and meanings exposed by all the main browsers differ. As a result, the browsing ecology becomes very challenging to safeguard. In three main browsers, we thoroughly examined more than a thousand security configuration choices, and we discovered that just 17 of them had names and meanings that were widely used. We outline the findings of our in-depth investigation in this publication. We also discuss Open Browser GP, a knowledge-based program that would let businesses set up extremely specific safe configurations for their information and operational technology (IT/OT) browsing ecosystem.

Fiaz Khan (2016) Global consumers can access scattered resources using cloud computing. Scalable design seen in cloud computing offers firms in several industries on-demand services. However, there are several difficulties with using cloud services. For various types of issues present in cloud services, many approaches have been presented. In order to address the issues with SLA, this study examines the many models for SLA in cloud computing that have been presented.

Garg, Radhika & Stiller (2015) There are several aspects to consider when deciding whether or not to embrace cloud-based technologies in a particular IT (Information Technology) environment. In order to properly deploy cloud-based services and assess their subsequent impact, it is necessary to identify key parameters that signal the performance of such services. The purpose of this study is to examine and identify the technological, financial, and administrative considerations that are important. This is exploratory study that entails (a) reading up on cloud computing and (b) doing various case studies with 17 companies who have already implemented or are considering cloud computing. This research also addresses the intricacy and interdependence of these aspects, since they are not independent of one another.

De Marco et al.,(2015) A legal document known as a Service Level Agreement governs the provision of cloud computing services (SLA). It is signed by both the client and the service provider after a period of negotiation, and both parties must abide by certain stipulations while the contract is in effect. Cloud services are widely utilized, but they are also often exploited, especially by cybercriminals. In rare cases, criminal acts may result in a breach of a contract's terms, even if the parties involved were unaware of the breach. A specialized system communicating with the cloud services and identifying the SLA breaches by analyzing the log files is one way to ensure better control over the SLA respect. As part of an automated technique for identifying SLA breaches, we develop a formal model to express the contents of such SLAs.

V. Binu& N. Gangadhar Binu (2014) When it comes to the service-oriented business model at the heart of Cloud Computing, Service Level Agreements (SLAs) are crucial. There must be a technique for negotiating and monitoring service delivery in SLA frameworks for cloud computing. For SLAs to be effective and widely accepted, they must be monitored in a reliable and safe manner. This article creates a Service Level Agreement (SLA) Framework that uses a third-party to negotiate and implement a secure system for monitoring compliance. We conduct a reference implementation of the Framework and utilize it to create a case study of a cloud-hosted video Trans-coding service provider. The case study demonstrates how useful the established framework is for developing SLAs for the kinds of complicated scenarios typically encountered while utilizing cloud computing services. The Service Level Agreement (SLA) service criteria used to define services are characterized via the lens of workload characterization.

S. Sen et al (2013) Cloud computing is revolutionizing the use and administration of IT by providing advantages such as lower costs, quicker innovation and time to market, and scalable on-demand application expansion. Our primary concerns are with cloud computing's security, privacy, and regulation. The future of cloud computing is briefly discussed, and some approaches are offered to addressing the issues that have been raised.

Andr´esGarc´ıa-Garce´ıa, et al., (2013), "SLA-Driven Dynamic Cloud Resource Management" presents the cloud compass framework could be dynamic adopted the for quality-of-service violations. The work contribution Cloud compass manages the resource lifecycle of SLA aware Pass Cloud platform.

A.S.Ferreira et al., (2013) presents a security monitoring architecture for the IaaS service model that is based on SLA. The monitoring strategy does not need the host to have a monitoring agent installed. Due to the method for checking the effectiveness of the deployed resources, the task can assist the management and execution of the Phase. Additionally, it aids in negotiation and deployment thanks to resource deployment based on security SLAs and security parameter specification thanks to a method for expressing security rules in SLAs.

Blasi et al.,(2013) the cloud market is rapidly evolving, resulting in the introduction of new services, methods for delivering services, and interaction and cooperation models between cloud providers and service ecosystems that utilize cloud resources. The rules of engagement for the participating businesses are laid out in Service Level Agreements (SLAs), which regulate the afore-mentioned interactions.

K. W.Ullah et al., (2012) This article introduces an automated technique for assessing cloud service providers' conformity with established security standards. It reveals the specifics of the safety measures provided by the cloud service.

J.Luna,et al.,(2012) “Benchmarking closed security level agreement using quantitative policy trees” presents a qualitative and quantitative method in security SLA between cloud provider and consumer. Evaluation security requirement based on Quantitative policy tree. The work contribution in specification and definition, negotiation in security parameter.

C.A.Silva,et al.,(2012)Evaluation of cloud resource provisioning according to hierarchical security metrics is proposed in "A methodology for management of cloud computing using security criteria." The GQM (Goal-Question-Metric) method is used to create the hierarchical metrics. Because of the security metrics that are created and can be stated in SLAs, Phase 2 is facilitated, allowing for service provision based on security needs and specification of security metrics.

The VEP (Virtual Execution Platform) is described by Y. Jegou et al.,(2012) Managing of apps under sla limitations on contrail," and is responsible for allocating IaaS resources according to the SLA's specifications. It is also VEP's job to keep an eye on the deployed instance over its whole lifetime. This effort contributes to Phase 2 through execution and management monitoring of the deployed instance's execution based on security SLA requirements.

Salman Baset (2012) With so much variation across cloud service providers, we wondered how these SLAs stack up against one another and how they need to be set for the future of cloud computing. In this article, we'll dissect a cloud SLA into its component parts so you may evaluate the terms offered by different public cloud service providers. According to our research, customers are responsible for detecting SLA violations and none of the assessed cloud providers provide any performance guarantees for compute services. As a follow-up, we offer recommendations for the future definition of SLAs for cloud services.

Nie, Guihua et al., (2012) More and more people are switching to cloud services because of the convenience and lower upfront costs associated with cloud computing. The parties to a cloud computing service should enter into a service level agreement (SLA). It has the potential to standardize the monetary terms of the partnership while also ensuring the highest possible standard of cloud service delivery. The authors offer a web service level agreement–inspired approach for cloud service agreements. Both a management and a coordination model are a part of this. The cloud service may be created, released, measured, evaluated, managed, and discontinued mechanically.

Stankova et al., (2012) We examine SLAs for cloud computing services, focusing on those offered by infrastructure-as-a-service (IaaS) providers and available on their own websites. The goal is to examine whether or not SLAs actually serve as instruments that boost trust. There has not been as much general adoption of cloud computing as there could be because many decision makers do not trust the technology or the providers enough to do so. By fostering confidence, we may speed up the development of cloud computing. The most important facets of trust are covered, together with the standard qualities outlined in SLAs. The next section presents a review of the SLAs that are actually used by IaaS providers and may be found on their websites. One conclusion is that few providers now make use of SLAs to their full potential as trust-building tools.

Takabi et al., (2011) Although the cloud computing paradigm is still under development, it has lately accelerated significantly. However, the main obstacle to its rapid adoption is security and privacy concerns. The authors of this paper discuss security and privacy issues that are made more difficult by the peculiarities of clouds and demonstrate how these issues relate to various delivery and deployment approaches. They talk about alternative strategies to deal with these issues, current fixes, and upcoming work required to deliver a reliable cloud computing environment.

J. L.Gracia et al., (2011) A security metrics framework for the cloud proposed a framework to develop security metrics for evaluation by cloud provider, taking into cloud service model and deployment. This work contributes definition and specification of security parameter in SLA Lifecycle.

Rajkumar Buyya et al., (2011) SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions presents architecture, vision and challenges of SLA orientated resource management in cloud computing. SLA oriented resource improve the system efficiency and minimization the SLA violation of service provider.

Mohammed Alhamad et.al., (2010) SLA-Based Trust Model for Cloud Computing advocated the novel trust model collaborate with the conceptual SLA framework for cloud computing Different domain of cloud service provider evaluate the reliable resource to domain of cloud user.

Artur Andrzejak et al., (2010) Decision Model for Cloud Computing SLA constraints proposed the probabilistic model for optimization of reliability ,monetary cost and performance for dynamic condition. The work contributes Cloud SLA constraint given resource availability for job compilation.

CHAPTER 3

Execution Of Negotiation Protocols And Sla-Resource Manangement In Cloud Computing Framework

3.1 An Approach To Negotiation Protocol Development And Use

This section offers a general solution to the issue, namely the creation and use of negotiation protocols. This section draws on the author's published works. As was stressed, there is market demand in cloud services with additional value that are QoS-based. This highlights the usefulness of Service Level Agreement (SLA) discussions, which enable customization of service requirements and the acquisition of necessary resources across the value chain. To reduce their liability, carriers already provide rigorous take-it-or-leave-it SLAs. Take-it-or-leave-it SLAs have been condemned by the European Cloud Computing Strategy as being unfavorable to users and have been replaced with phrases like data integrity, confidentiality, ownership, and service continuity. The US government's cloud computing policy makes similar requests.[36]
This research frames agreements as flexible business models that can leverage the ever-changing nature of clouds to sell individualized services that offer value for customers. A negotiating protocol is a predetermined procedure for making a proposal of a certain value. Additionally, the European Commission recommended that service providers "allow runtime adaptability via dynamic automated SLA (re)negotiation procedures" in its 2013 exploitation report on cloud computing SLAs. Here, SLA talks bolster the on-demand acquisition of cloud services. Advantages are compounded when a SLA is negotiated. For instance, it may be necessary to modify your business plan or account for swings in demand by renegotiating your present SLAs. Negotiation strategies are used as business models to distinguish various services. For instance, by allocating spare resources (referred to as spot instances) through a bidding process, Amazon developed a distinct cloud market for fault-tolerant applications. EBay is a similar platform for exchanging products that operates on an auction model. As long as price is the only topic of discussion during negotiations, these processes lack the idea of SLA.

This encourages the use of multi-round SLA agreements as a workable market segment for cloud or cloud-based services.[37]

Specific market niches are targeted by protocol-governed electronic marketplaces, such as spot instances, which don't replace on-demand or reserved instances markets. Additionally, it is true that no one protocol can address every possible negotiating circumstance. It is suggested that eventually, a system of marketplaces employing various protocols would evolve rather than a single protocol or central marketplace. But creating and using several protocols is still a difficult task. The majority of earlier efforts have only used one protocol. Diversification is hampered by this rigidity, which is made worse by three problems. First, research on protocol development has been quite subjective, focusing on the steps of specifying, validating, and implementing in split or case-based ways. Second, business configurations frequently mix up the architecture of the protocol and prevent its reusability. Thirdly, it might be challenging to guarantee that the protocol will be followed exactly and consistently by all participants. In keeping with that, this part offers a general-purpose framework for initiating and carrying out negotiation procedures. Our method seeks to coherently amalgamate essential ideas from the design, verification, and implementation phases of protocol development. It is based on a comprehensive examination of the previous art and needs from the SLA@SOI use cases. The aforementioned features are clearly covered:

- Creating protocol specs that can be tested for correctness and other features.
- Domain-specific hooks for parametric setting of the protocol to enable tailoring of negotiation sessions via code flexibility.
- Interoperable protocol execution without hitches.

The following are some of the specific contributions made to address these issues and fill in some of the gaps in the state of the art:

1. One, a methodology that can be used to the whole process of designing and deploying negotiation protocols, which treats them as objects that can be independently verified and interpreted by machines. As an example, we introduce the Simple Bilateral Negotiation Protocol (SBNP), a multi-round, flexible protocol used in SLA@SOI use cases and seen in action in the context

of the cloud value chain use case.

2. The ability for geographically dispersed parties to use the same negotiating platform to carry out the same protocol with the same results.
3. A verification-based assessment demonstrating the protocol's effectiveness if value chains are coordinated on a centralized marketplace.

3.1.1. SLA management framework

In the course of working on SLA@SOI, a framework for Generic SLA Management (GSLAM) was developed. The GSLAM platform automates SLA management across IT infrastructure. In Fig. 1, we see a simplified version of the overall architecture of the GSLAM framework. Generic parts are included within the dashed line; the remaining parts must be tailored to the provider domain by implementing the appropriate interfaces.[38]

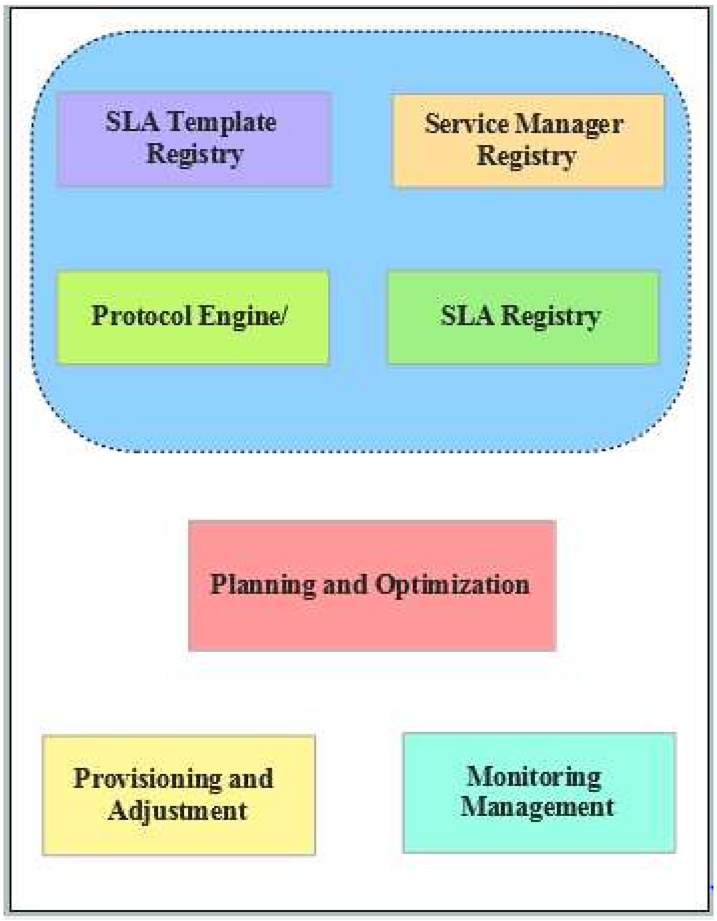


Figure 3.1.: Framework of Generic SLA Management

The Protocol Engine, a platform for negotiations, is a crucial component of the GSLAM implementation. Section below provides an explanation of its structure and operation. A storage for existing SLAs is provided by the SLA Registry component. Information about the provider's service landscape, licensing, dependencies, and resource requirements across service tiers is kept in the Service Manager Registry. The Planning and Optimization subsystem acts as the pilot when putting into action decision models for negotiating or consolidating and managing resources.[39]

In a nutshell, the framework facilitates effective communication, collaboration, and platform development and maintenance amongst geographically separated partners that are bound by a service agreement. To do this, individual modules must locate one another and establish communication channels through their respective SLA Managers. To locate and disseminate a provider's SLA templates to interested parties, the SLA Template Registry employs a broker component of it’s publish/subscribe based marketing system. This results in the open cluster of widely spaced SLA Managers seen in Fig. 3.2.

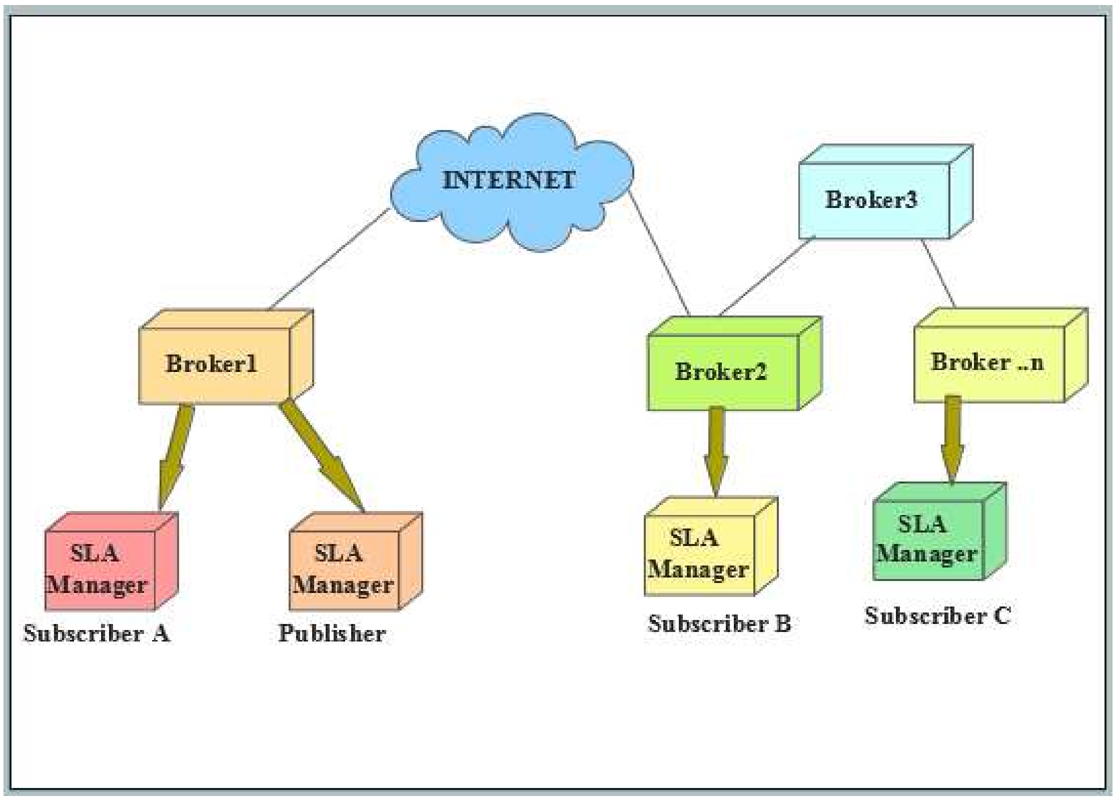


Figure 3.2.: SLA Template Discovery by SLAM Advertisement System

3.1.2 Development of Negotiation Protocols

We express the following prerequisites for protocol design based on our usecases and an extensive examination of pertinent literature.

- R1 Interaction and negotiation scenario: SLA dependencies, such as the delivery of infrastructure or software from other providers, may be present in service proposals. A conversation is intended to include parties in this negotiating scenario.
- R2 Visual descriptions are simple to understand, but formal descriptions clear up any confusion.
- R3 Functional and non-functional aspects: Non-functional characteristics like state space development and inconsistent states must be checked for in a protocol. Similar checks must be made for accuracy requirements.
- R4 Implementation: The interaction is carried out in machine executable form according to a confirmed protocol.

We propose a systematic protocol development lifecycle with four stages—modeling, verification, implementation, and general execution—to fulfill these needs everywhere. In order to illustrate these steps, we propose the SBNP protocol to be used in the negotiation.[40]

Situations under which negotiations take place: The success of a negotiation depends on the characteristics of the scenario being discussed. Our situation was based on the cloud value chain use case. The common practice of chaining among SaaS, PaaS, and IaaS providers simplifies the purchasing process for the consumer. To counter this, the PaaS provider has the ability to dynamically expand the PaaS cloud by, for instance, incorporating more (or removing existing) virtual resources from participating IaaS providers in the PaaS federation.

Cloud value chains offer a practical setting in which to negotiate service level agreement (SLA)-based procurement over the whole life cycle. The ultimate (SaaS) customer has QoS concerns. As illustrated in Fig. 3.3, a PaaS provider's service level agreement (SLA) duties relate to the quality and number of PaaS resource containers that the PaaS provider must make available to its customers. At each tier, negotiations take place to resolve any SLA issues and prevent over- or under-procurement of resources.[41]

3.1.2.1.Modelling

Sequence diagrams have been used to represent interaction behaviors in protocol specifications for ICNP and WSAG/N. Sequence diagrams have the drawback of only capturing incomplete activities. Though vital, seeing a relationship is insufficient. Unambiguity may be removed using a modeling technique that can be formally described, and automated verification can be used to weed out inconsistencies and undesirable behaviors.[42]

As a result, we employ Communicating Finite State Machines (CFSM), which may abstractly address these needs to variable degrees. Bilateral protocols with "sender" and "receiver" process roles are of special interest to us. We now use CFSM to demonstrate how these processes interact with one another.

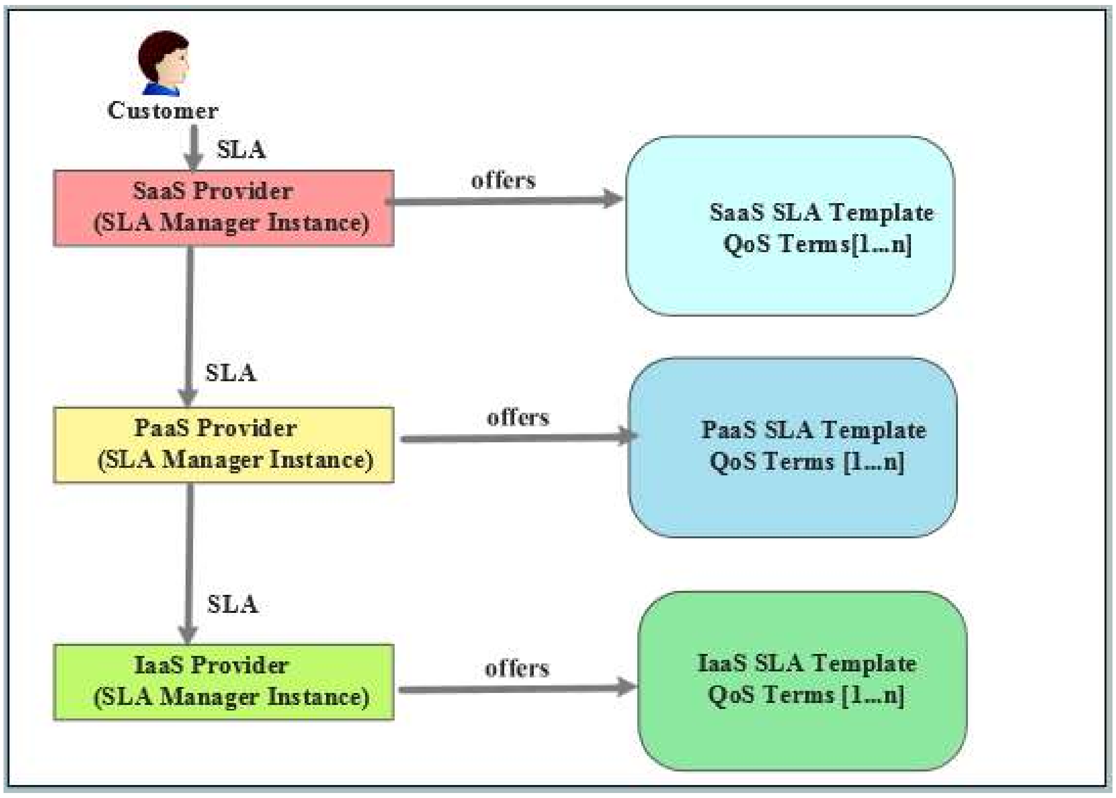

Customer
SLA
SaaS Provider (SLA Manager Instance)
offers
SaaS SLA Template QoS Terms[1...n]
SLA
PaaS Provider (SLA Manager Instance)
offers
PaaS SLA Template QoS Terms [1...n]
SLA
IaaS Provider (SLA Manager Instance)
offers
IaaS SLA Template QoS Terms [1...n]

Figure 3.3.: Negotiation scenario and SLA dependencies

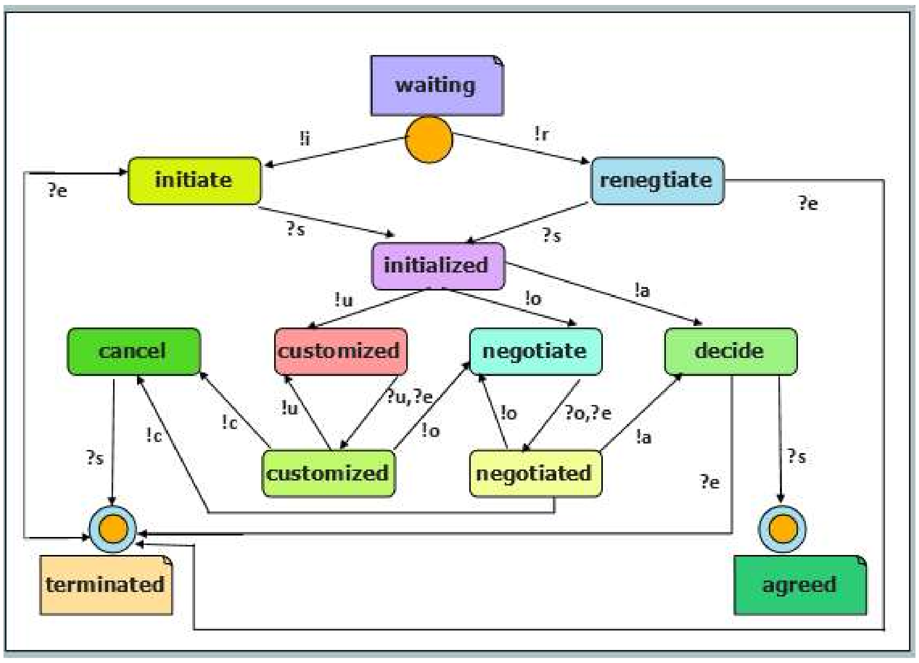
The very first thing that happens throughout the modeling process is the discovery of feasible states. In order to precisely characterize an interaction, we propose a finite set S = "waiting, initiate, renegotiation, initialized, customized, negotiate, negotiated, decide, cancel, terminate, and agreed." Sender and receiver CFSM for SBNP are depicted in Fig. 3.4(a) and Fig. 3.4(b), respectively (b). There is a clear distinction between the request state and the response state, and messages trigger transitions between both.[43]

Finding the messages that can be sent to the states is the second step. In response, we offer a list of messages, which are listed in Table 3.1 along with an explanation and their alphabetic abbreviation. The negotiation interface that results from messages has a crucial relationship with it. The claim is that the suggested states and messages can be expanded or reused to create new protocols.

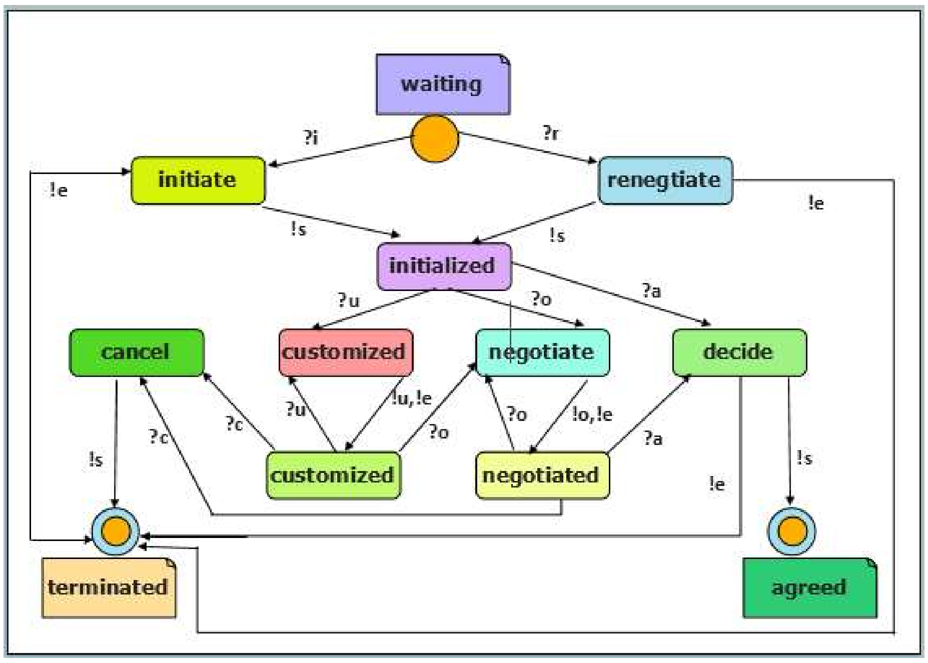
The third step is where protocols' parametric configuration is dealt with. Six parameters were identified; they are listed in Table 3.2. This allows for the setting of various parameter values for various negotiations.[44]

For this reason, we provide a partial CFSM formalism for SBNP, a two-way (client-server-like) protocol. We define SBNP as a network of CFSM(s) A such that each machine in this network represents one negotiating agent, given the set of states' S and a non-empty finite set of negotiating agents A = {a0, a1,..., an}. Agent and machine mean the same thing in this setting. This means that every a ∈ A

$$A = (A_a, I, F)$$



(a) Sender



(b) Receiver

Figure 3.4.: Simple Bilateral Negotiation Protocol (SBNP)

Table 3.1.: Protocol States, Messages and Alphabets

State	Message	Alphabet	Describition
Waiting	-	-	Wait for negotiation commencement request
Initate	intiateNegotiation	I	Request to initiate negotiation
Renegotiate	renegotiateAgreement	r	Request to renegotiate an existing SLA
Initialized	-	-	Response state for initate and renegotiate
Customize	customizeParameters	u	Resquest to modify protocol's default parameters
Customized	-		Response state for customize
Negotiate	Negotiate	o	Request that provides an SLA offer
Negotiated	-	-	Response state for negotiate
Cancel	cancelNegotiation	c	Resquest to gracefully cancel negotiation session
Decide	create Agreement	a	Request create an SLA of proposed(final) offer
Terminated	-	-	Response state that ends negotiation unsuccessfullt
Agreed	-	-	Response state that ends negotiation successfully
-	successfully response	s	A positive response message
-	Unsuccesful response	e	A negitive response message

Table 3.2.: Protocol Parameters

Parameter	Descripition
Process Timeout	Life time of negotiation process
Customization Rounds	Rounds for fixing protocol parameters
Negotiation Rounds	Rounds for exchanging offers
Max Counter Offers	Offers sent as response to received offer
Optional Critique On Qos	Critique on term value e.g.., increase,decrease,change or acceptable
Quiescence Time	Inactivity time among negotiating agents
Chain Length	Allow length of negotiation service chain

$A_a = (S_a, \rightarrow)$ is a finite state machine

S_a is a set of local states

$\rightarrow \subseteq \times$ Act a $\times$ Sa is a set of local transitions

$Act_a \subseteq Act$ is a set of local actions

I is a non –empty set of global final states

F is a non-empty set of global final states

The set Act = {send, receive}is a collection of A's linguistic behaviours employed by its local transitions to convey A_a from one state to another. All interactions between agents occur via FIFO channels, where messages arrive and are processed in reverse order of receipt. Agent ai and a_{i+1} communicate over a channel c = (a_i, a_{i+1}) and c' = (a_{i+1}, a_i) where $1 \leq i < n$. To convey message m from agent a to agent b, the action send has been specified a_{i+1} as $Act_a^!$= (c!m). The analogous action, receive, is defined at agent ai+1 to pick up the old message as$Act_{a+1}^?$ = (c?m). Similarly, send is defined for agent ai+1 as $Act_{a+1}^!$= (c'!m).For Agent A, the equivalent send is defined as$Act_{ai+1}^?$ = (c'?m). Then, Ch = c, cj is a set of outgoing and incoming channel per agent-pair. The message $m \in \Sigma$ where $\Sigma = \{i, r, u, o, c, a, s, e\}$ consists of a group of letters used for sending and receiving.[45]

3.1.2.2 Interactions and Marketing Enablements

The structure of SBNP interactions is depicted in Figs. 3.4(a) and (b). Until a negotiation session is formed, the receiver process (the provider) remains in the waiting state after the sender process (the customer or the provider) invokes the commence Negotiation or renegotiate Agreement message. The negotiate message makes it possible to make many (non-binding) offers and counteroffers. By default, a negotiation platform will enforce a provider's preferred settings for negotiation parameters. Through the customize, SBNP also offers an optional interaction. Prior to the exchange of offers, parameters messages are sent between the parties to adjust parameter values. The cancel Negotiation message command can be used to end a negotiation. The final (binding) SLA offer is submitted using the create Agreement message. If the recipient agrees to the terms of the SLA, the agreement is generated and sent back. To further understand how SBNP facilitates a negotiation (value) chain, consider the hypothetical situation shown in Fig. 3.5. [46]

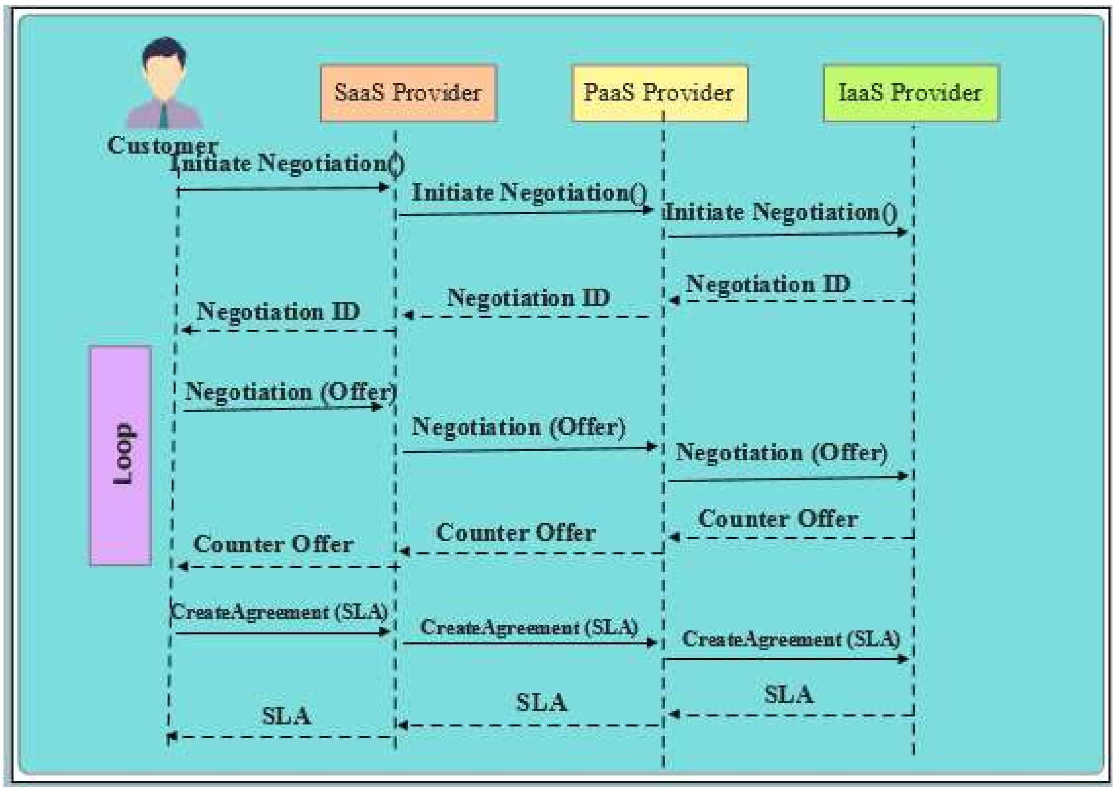


Figure 3.5: Negotiation Chain enabled by SBNP

A single operation may include several contacts. For instance, interactions including a single offer, many rounds, and customized multiple rounds are supported by SBNP. Interaction design reflects the service offer. SBNP performs effectively for ideas that only require one stage. However, identical procedures may be used to design protocols that enable multistage proposals. Consider a SaaS-IaaS proposition. After the first step, in which IaaS resources like storage are addressed, a second stage is optionally done to negotiate QoS for file synchronization and management services, such as own Cloud (SaaS), to be delivered over bought storage. Afterward, based on whether each negotiation step is optional or necessary, special service proposals can be developed. The development of service proposals using SLA aggregation techniques has been proposed. These may be utilized to develop novel negotiation procedures that allow for the inclusion of both the consumer and the supplier in the resolution of aggregation points.[47]

3.1.2 .3. Verification

As has already been mentioned, protocols must be validated before use. We use propositional dynamic logic as opposed to Spin model checker. If you want to use Spin, you'll need to have a protocol model that's been expressed in PROMELA abstract syntax. PROMELA provides in-built primitives for encrypting processes, states, channels, and messages sent between any numbers of processes. Based on the PROMELA description, Spin constructs the protocol graph and exhaustively explores all of its edges to see whether the correctness behaviors specified by LTL (linear temporal logic) attributes are respected or disregarded. Due to these characteristics, Spin may be used to validate not just SBNP but any CFSM protocol model. Considering the breadth and depth of the subject matter, we are confined to discussing only three specific contributions that have been overlooked by the existing literature. [48]

1. Spotting problems with the procedure before it's put into action or during operation.
2. Assessing the development of state space.
3. Establishing correctness properties in LTL form.

3.1.2 .4. Error Detection

No inconsistencies, such as undefined receptions or acceptance cycles, were found during SBNP's spin verification run. Progress cycles are defined for periods of negotiation and customization, which indicates the existence of a limit in practical application. As a result, Spin does not flag cyclic faults. SBNP is a secure protocol since there are no deadlocks or live locks.
However, the controller component, which powers the negotiation platform, is responsible for some control activities in negotiation systems. Retransmitting a message if it is not promptly responded to is standard procedure at this level, as long as the allotted time for negotiations or the number of rounds are not exceeded. Through non-determinism, such control scenarios may be simulated via PROMELA/Spin based verification. This powerful feature enables the detection of flaws in the planned runtime use of the protocol. A transition (referred to as timeout) was added to the SBNP verification program using this non-determinism. Using this change is similar to what a timeout would accomplish in a real system; the message is resent. Timeout transitions

like this one are shown in Fig. 3.6, which depicts their usage during the sender's negotiate phase, while waiting for a response.

As soon as Spin tested the new protocol model, it found deadlocks. Spin independently confirmed the error's specific nature. It's possible that the new change may cause messages to be repeatedly resent, leading to a backlog at the receiver. A single (late-arriving) wrong response to an earlier offer may force the receiver to transfer to the terminated state while it is still active, prompting the sender to change to the decide state. And if the receiver accepts the late agreement request after the sender has already terminated, the ensuing SLA will be unilateral. This kind of inconsistency (invalid final states) is unacceptable. Therefore, in this study, "Process Timeout" was chosen as a more secure alternative, with request level quiescence at the negotiation states being optional.[49]

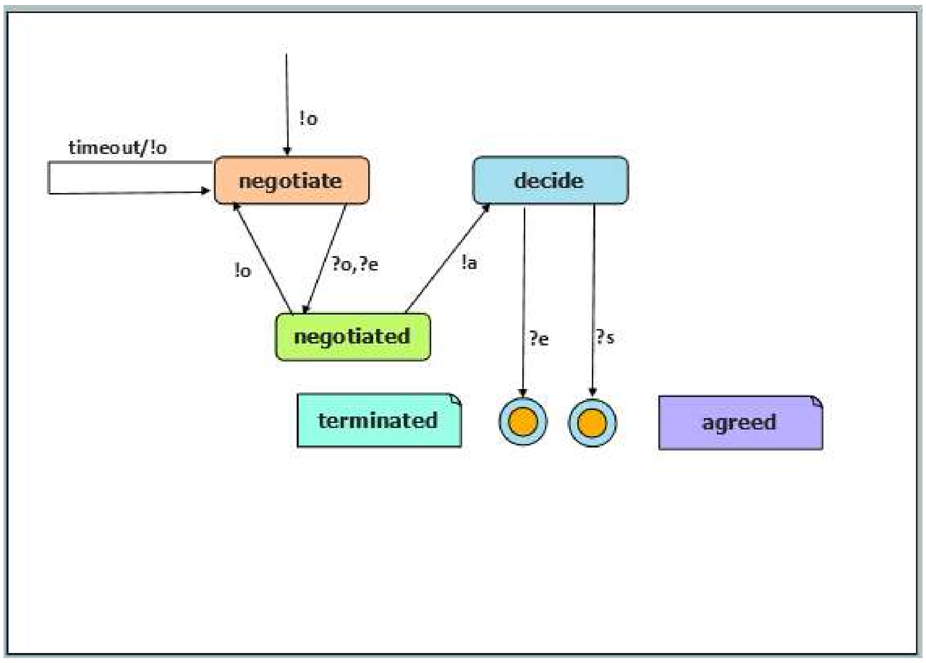


Figure 3.6.: Timeout at Sender

If a request or answer is lost, the machine will continue in its present state until the "Quiescence Time" option expires, at which time the communicating parties will cease the session willingly to prevent inconsistencies.

3.1.2 .5. Evaluating Protocol Scalability

While the SLA@SOI architecture is primarily intended for decentralized bargaining groups, its recommended negotiation platform may also be used to set up centralized marketplaces like eBay or Amazon. In contrast to current systems, the proposed work centre on linked negotiations, in which one provider agent's interactions with clients often spark off further conversations between that agent and other registered providers in the marketplace. In this case, determining the scaling requirements of a protocol is essential for dealing with memory issues. Keep in mind that a CFSM network is built up of paths that FSMs take to communicate with one another (representing client and provider agents). In this case, the term "scalability" of an SBNP refers to the fact that the number of states in a CFSM network will increase as more agents join the negotiation chain (see Fig. 3.5). Expanding the state space in the worst possible way involves.[50]

$$\prod_{i=1}^{N}\left(|\#\text{FSM states}|\prod_{Y}^{y\in V}|dom(y)|\right)\prod_{j=1}^{K}\left|dom(c_j)\right|^{cap^{c_j}}$$

According to our experiment conducted with SBNP's PROMELA program, the values of these variables have been adjusted accordingly. Spin uses this to model a series of negotiations in a chain. The software may make use of the variable set V to implement the PROMELA abstract syntax protocol automata. For the purpose of verification, these parameters are not included in the graphical (CFSM) model. The number of links in the chain, N, is the number of agents. States in the SBNP transmitter and receiver automatons, abbreviated "FSM states," number 12. As a single variable, y V, was employed for both transmission and reception.

Since the starting agent in a negotiation chain is simply concerned with recording whether or not a response was effective, its dom(y) is 2. In order to hold the six potential messages in SBNP and the two possible answers, the dom(y) for agents midway down the chain must equal 8. Finally, the agent at the end of the chain has dom(y) = 6 since it has no further processing requirements beyond storing messages. Each of the K channels has its own unique domain, denoted by dom(cj). Each agent-pair has two channels, with dom(c) = 6 for outgoing channels carrying the 6 potential messages and dom(c) = 2 for incoming channels carrying the 2 possible answers. The term "capacity per communication junction" (capcj) is used to describe the maximum amount of messages that may be exchanged in a certain period of time, such as a single negotiation cycle.

Spin distils state space by using partial order reduction to generate only reachable protocol states. In our experiment, Spin generated from 139 to 1363 states for N = {2, ..., 10} agents exchanging a single offer.

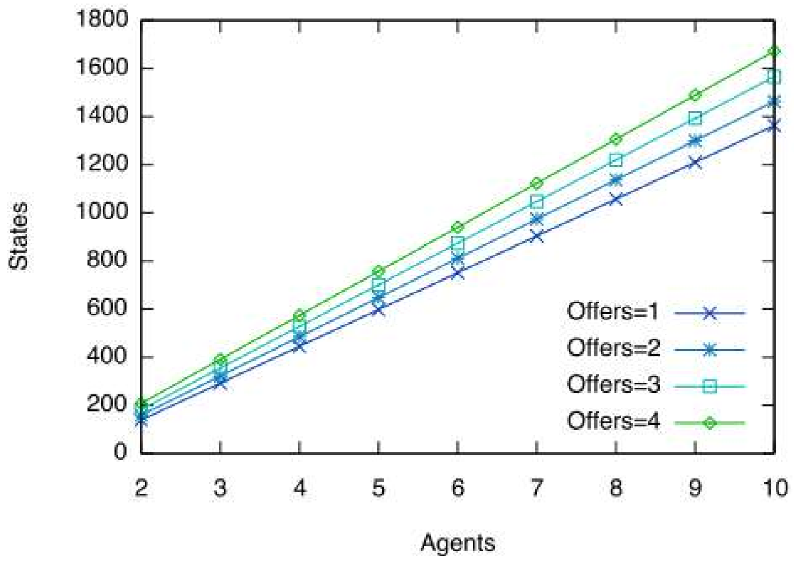


Figure 3.7.: Growth in States(E.Yaqub et al.,2014)

When trading 2, 3, and 4 offers, these rise from 159 to 1463 states, 182 to 1566 states, and 208 to 1672 states, respectively. This increase is controllable, and the graph in therapy. Additionally, it demonstrates the necessity of setting protocol settings like "Chain Length" and "Max Counter Offers" to minimal levels. In negotiation chains, linearly or sub-linearly scaling protocols are chosen since negotiation tactics increase the computing burden.[51]

3.1.2 .6. Functional Correctness

It is the negotiating context that informs the functional needs (s). Requirements formalization is not a simple task. Therefore, we illustrate this with an example, which provides a useful method for validating other protocols as well. Our case (Figs. 3.3, 3.5) necessitates two SBNP capabilities:

1. To prevent over committing, it is necessary to build dependent SLAs at each stage of the negotiation chain while drafting a SLA with the client. The opposite is true: over-procurement may be avoided by not establishing dependent SLAs if a SLA is not established with the client.

2. No more than one SLA may be established between any two parties in a discussion.

Requirements are formalized in LTL as liveness behaviors or safety behaviors. State transitions in time are stated concisely as LTL characteristics, making up behaviors.

As a first step, the verification software stores a flag whenever a particular condition is reached. A boolean condition written in flag form is called a proposition (s). The validity of a hypothesis may be established by a search. Property may be decked up in security garb if desired (preceded by ⊡ symbol, which tests its universality) or as liveness property (preceded by ♦ symbol, which tests its eventual permanence). Combining propositions with logic and modalities like or allows us to investigate the occurrence of intricate sequences of states conjunction (∧), disjunction(∨), negation (¬), implication (→) and until (∪).Dwyer's property specification pattern templates make this onerous effort more manageable by allowing you to transfer your existing, well-understood understanding of system behaviors into LTL properties.

Our primary criteria have a precedence and response structure. The concept of precedence states that the occurrence of one state (the cause) is required for the occurrence of another condition (the consequence) to take place. Reaction holds that the presence of a cause prompts the emergence of an effect. Precedence The cardinality to precedence chain pattern describes the relationships between a single cause and several effects. The Response Chain Pattern applies cardinality to Response in a way that's similar to the Response Chain Pattern, which shows how one stimulus may have several responses (effects) and vice versa. When using these patterns, identifying the states of cause and effect is essential. The circumstance that leads to the first agreement is sometimes called the "cause." The reason for this is because under SBNP negotiation chains, the last provider in the chain is given the authority to draught the first agreement.[52]

The l_agr flag indicates that an agreement was made at the recipient end of a two-party SBNP discussion, while the s agr flag indicates that an agreement was created at the sender end. If the PaaS provider sets the flag l_agr, indicating that they want the IaaS provider to make an agreement, then there is grounds for SBNP chained discussions. To indicate that the agreement was made by the PaaS provider at the behest of the SaaS provider, the flag m agr is set. When a client requests a new agreement with their SaaS provider, the flag s_agr is set to indicate that the provider has fulfilled that request. This is how we get to our LTL features.

The concept of "Precedence" is not strictly enforced in LTL theory. This simple rule allows causes to take place even if there are no following consequences. Reaction makes uncaused consequences possible. Response is the opposite of Precedence and is used to establish causation. Properties like Precedence and Response work hand in hand to pinpoint behavior.

3.1.2 .7. Implementation

Implementation is the pinnacle of protocol development. Using a format that is informative, simple to encode, and capable of generic execution with little to no recompilation presents issues in this context. The body of knowledge on using negotiation procedures is scant. The WSAGN standard has a Java implementation that is offered by the WSAG4J framework. The protocol is implemented in this case as imperative code, which makes it difficult to change or expand. promotes the use of XML to implement protocols. Although XML offers the advantages of an explicit and expressive syntax, achieving protocol-generic execution requires expanding the platform to parse and handle each additional protocol.

We contend that a protocol that precisely describes finite state machine (FSM) interactions should exist as a self-contained, executable product. This implies that a protocol may be made available for download as a file, preventing inconsistent implementations of a single protocol definition. This requirement is strongly contested. By executing many protocols in a single execution environment, this also results in a protocol-generic footprint in numerous marketplaces.

Declarative rules properly satisfy these technological criteria. FSM semantics like transitions and guard conditions are amenably mapped by rules into executable logic that may be simply expressed as "IF-THEN" clauses. Because the majority of the platform's processing complexity may be delegated to the rules rather than the platform itself, we place emphasis on the executable nature of rules. By doing this, it is prevented that the protocols in the platform code get cluttered or that XML documents become challenging to understand. We take use of the rules' adaptability to accomplish generic execution.[53]

We specifically chose the very powerful Drools rule syntax, which enables declarative Java object composition. In the IF section of the rule, our rule encoding approach links an input event to a state. This serves as a prerequisite. If accurate, the THEN section takes a specific action. A feedback command (setProcessed) indicates whether the event that the current state is processing resulted in a positive or negative response. At the conclusion of the rule, the event is withdrawn from working memory. In this manner, rules create transitions between state objects in the working memory and match permitted events with states.

To illustrate how SBNP is implemented using rules, we now give three condensed rule snippet examples. On receiving an offer, initialization state transits to the negotiation state, as shown in Fig. 3.8. Figure 3.9 depicts a rule that carries out

```
Rule Initialized_To_Negotiate_Transition
 1: IF
 2: initializedState : State(name==INITALIZED);
 3: event : Event(name==OfferArrivedEvent);
 4: THEN
 5: initializedState.remove();
 6: insert(new State(name=NEGOTIATE));
 7: event.setProcessed(true);
 8: retract(event);
```

Transition Rule

In the event that an offer is received, a guard condition is applied to the negotiation state. If there have been no more than the specified number of rounds of negotiations, the rule is executed. The round counter for negotiations is then increased. Lifespan constraints in negotiations are enforced by the rule shown on Fig. 3.10. To determine whether the allotted time has elapsed, it employs the program's internal evaluation mechanism. If that's the case, the platform will conclude the negotiating session with a negative response and an explanation. This rule is not limited to any specific occurrence, but rather applies universally (line 4). A salience (priority) value may be set on rules to fix execution order when several rules share the same event and are both eligible for execution. Since the time for negotiations has expired, the rule with the greatest salience (here, 5 in our example) will now apply.[54]

Rule Guard_Negotiation_Rounds

```
1: IF
2: negotiateState : State(name = = NEGOTIATE, rounds ≤
   maximumNegotiationRoundsAllowed);
3: event : Event(name = = OfferArrivedEvent);
4: THEN
5: negotiateState.rounds++;
6: negotiateState.update();
7: event.setProcessed(true);
8: retract(event);
```

Guard Condition Rule

Rule Control_Negotiation_Time

```
1: salience 5
2: IF
3: params: Parameters( eval(currentTime() ≥ (startTime +
   processTimeout) ) );
4: event : Event();
5: THEN
6: event.setProcessed(false);
7: event.setReason('Negotiation time out');
8: retract(event);
```

Negotiation Time Rule

In addition to the implementation of the protocol, domain-specific business rules are established, such as rules for defining the values of protocol parameters at the outset of negotiation or rules for defining the customization states. The ability to tailor protocol parameters according to predefined business rules is thus enabled. At the outset of a negotiation, the rule engine receives an object representing each participant's profile; this object is used by one of the rules to determine a ranking. On the basis of her profile, the provider's business policy rules may assign values to protocol parameters like negotiation time and the number of rounds of negotiation. This enables a negotiator-specific trade-off between negotiation complexity and convergence probability. Such a business rule is demonstrated in Fig. 3.11. Similar to how customers may be favoured or negotiations can be refused entirely by writing whitelisting and blacklisting rules.[55]

```
Rule Customize_Protocol_Parameters
 1: IF
 2: initializedState  : State(name = =INITIALIZED);
 3: event : Event(name = = CustomizationArrivedEvent);
 4: THEN
 5: initializedState.remove();
 6: insert ( new State(name=CUSTOMIZE));
 7: if( partnerProfile.getRank = = DESIRED)
 8: insert new parameter(NEGOTIATION_ROUNDS=100));
 9: else
10: insert(new Parameter(NEGITIATION_ROUNDS=10));
11: event.setProcessed(true);
12: retract(event);
```

Business Rule

3.1.3. Negotiation Platform for Protocol Execution

To facilitate direct negotiations amongst themselves, SLA Managers use the Protocol Engine's SLA template. A webservice end point reference negotiation interface is used to deliver messages to the Protocol Engine (EPR). Methods discovered during modelling provide the basis of the negotiating interface seen in Fig. 3.12. (see Table 3.1). Their role is described in summary form below.

initiateNegotiation: Make a formal request to begin bargaining. Each time a session begins and ends, a unique ID is created and returned.

renegotiateAgreement: Make a formal request to modify an existing service level agreement. Each time a session begins and ends, a unique ID is created and returned.

customizeParameters: Specify suggested values for the protocol's configuration parameters and submit the request.

negotiateOffer: Make a SLA proposal. The platform will then forward the proposal to the POC section. POC is able to make counter offers that are then sent back.

createAgreement: If the presented offer is satisfactory, please request that we enter into a SLA. The POC component makes the final call after receiving the offer from the platform.[56]

cancelNegotiation: Request to cancel negotiation

Negotiation Platform for Protocol Execution

```
UUID initiateNegotiation( SLATemplate offer)
 UUID renegotiateAgreement (SLATemplate offer)
 Parameters customizeParameter ( UUID negotiationID, Parameters params)
SLA Template[ ]negotiateOffer (UUIDnegotiationID, SLATemplate offer)
SLA  createAgreement( UUID negotiationID, SLATemplate offer)
cancelNegotiation(UUID negotiationID)
```

Negotiation Interface

To ensure maximum adaptability, we made certain design decisions on the isolation of problems. These features make the Protocol Engine (short PE) stand out from competing systems:

- The disconnection between the protocols and the host system.

- Keeping bargaining tactics and customs apart.
- The disassociation of the platform from any one SLA standard (s).

The PE's internals are seen in Fig. 3.13. At the time of the initiate call, the Negotiation Manager looks for the appropriate protocol in the SLA template parameter and then requests that the State Manager component create a negotiation session in memory.

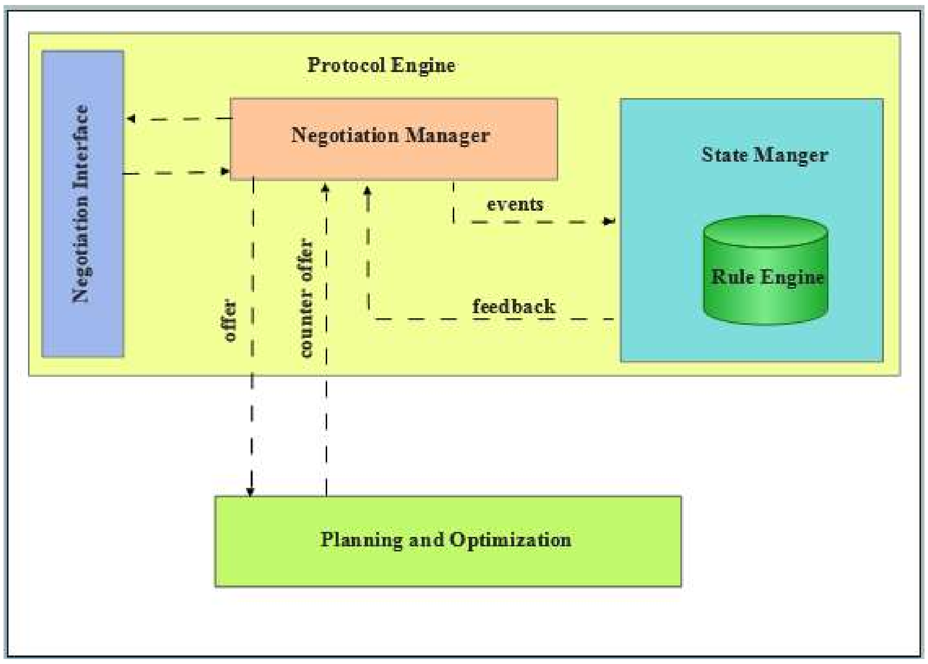


Figure 3.8.: Design of Protocol Engine

The State Manager uses a Drools rule engine instance, which loads the protocol file into memory for processing. This is how the associated state machines are started up. In the last section, we saw how the events sent from PE to the State Manager, which are then stored in Drools' working memory, are employed in the rule encoding scheme. These events correspond with messages sent and received. Forward-chaining is the method by which Drools executes all applicable rules once an event happens or an object is created or modified as a consequence of the execution of a rule. This platform improves Drools' already powerful rule management features. Due to the lack of tight coupling between the protocol and the platform, the protocol may exert sufficient control on negotiation to the point where specific fault causes are sent to the caller in the form of rules.[57]

Also, for generic execution, a strict divide between protocol and strategy was required. Therefore, we do not overburden the regulations in order to process an offer and generate a counter offer (s). The negotiating strategy is implemented by a Planning and Optimization (POC) subcomponent (see Fig. 3.13). In this manner, several combinations of protocols and tactics are feasible, allowing for the sale of services through various negotiation protocols and methods.

We employ a SLA model to express SLA templates and SLA offers, as can be seen in our negotiation interface. However, it was crucial to retain the SLA Manager framework independent of any particular SLA standard. Through the use of a Syntax Convertor component, compatibility with existing or upcoming standards is supported. For example, the Syntax Convertor was able to communicate with WSAG-based clients by 1) implementing a web service endpoint for the WSAG interface, and 2) transforming calls and arguments from WSAG's XML format into our SLA model's Java format for transmission to the PE.

The platform provides access to a set of control interfaces and a set of negotiation interfaces that may be used to set up business rules that stand in for institutional policy. Techniques such as these are included into the interface:

- setPolicies: Create the set of guidelines for the policy. The rule set may be new or it may replace an existing one.
- getPolicies: Get the policy rules list and send it back.

Changes to business rules (like those shown in Fig. 3.11) that rank, whitelist, or blacklist clients may be made on the fly with access to the rule engine, negating the need to restart the negotiating platform or recompile the code.

There was a comprehensive proposal for resolving the bargaining problem. It has been argued that Service Level Agreement (SLA) agreements provide a flexible and usable business model and that specific (variable) protocol serve as provider differentiators or a fertile ground for new markets centred on value-added service offerings. One of the contributions was a systematic method for constructing machine-executable negotiation protocols. Bilateral negotiating procedures were developed using the same method. In addition, a standardized framework for expressing protocol rules was developed, and a common negotiation platform was developed to implement those rules. The protocol, strategy, and specific SLA standard worries are separated on the platform, making localised adjustments possible. The protocol and platform have been implemented in a wide range of use cases by both the SLA@SOI and Contrail projects (of which the author was not a part of). This shows that the proposed solution can be used to many different situations.

The proposed approach was argued to provide a protocol-generic foundation in a market structure where SLA agreements are used to acquire cloud-based resources or services. Although single-staged negotiations (albeit with numerous rounds) were addressed, future possibilities include multi-staged protocols that may be used to develop packaged IaaS, PaaS, and SaaS service offerings based on required or non-mandatory stages of negotiations.[58]

3.2 Management Of Resource In Cloud Computing Of Sla-Aware

This section offers SLA-aware resource management in cloud computing as a solution to the issue.

The increasingly common Platform as a Service (PaaS) cloud service delivery paradigm is especially taken into account in this study. PaaS increases the level of abstraction on infrastructure resources to provide software platforms as a service. This is accomplished by offering an ecosystem for software services that enable the cloud and govern their lifespan using runtime controls. By using sophisticated automation, PaaS conceals for the user the management difficulties of the underlying IaaS, hastening the creation of cloud-based apps. The PaaS is gaining commercial momentum as a result of its enormous potential. However, the IaaS model's effective management of virtual machines has received a lot of attention in studies to date. Operating system (OS) level containers are the foundation of the PaaS deployment architecture. By safely hosting various modules of numerous program on a common OS, this raises usage. Micro-services, a novel approach to software development made possible by PaaS, enable fine-grained administration of service components as portable and light-weight virtual containers. Contrary to virtual machines (VMs), containers may be efficiently scaled, relocated, or positioned next to other services or data sources depending on SLA needs. SLA proposals with additional value can take advantage of this.

However, the author's PaaS study found that on-demand procurement, unexpected workloads, and auto-scaling lead to rapid increases and decreases in the number of automatically provisioned containers and the unintentional usage of the underlying machines. This is a major problem on a global scale. For instance, Google recently said that it operates all of its services, including email and search, in containers and that it deploys over 2 billion containers every week across its international data centre. OpenShift Online, a public PaaS from RedHat built on Amazon EC2, now hosts more than 2.5 million applications. Therefore, the primary challenge for a PaaS provider is to regularly plan and optimize the placement of containers on computers in order to meet SLA commitments, optimum resource utilization, and use the fewest amount of machines feasible.

On the other hand, allocations fulfilling merely resource capacity needs are impractical due to service-driven restrictions regarding containers and geographical limits regarding machines. This relatively new "Service Consolidation" problem is NP-hard since it is a variation of variable-sized multi-dimensional bin-packing. Additionally, it is important to consider other criteria, such as estimated SLA violations, energy consumption, migrations, resource contention, utilization, total machines used, and resource contention when evaluating feasible solutions. The latter creates significant economic and environmental issues, with data centre accounting for 1.1% to 1.5% of worldwide energy consumption in 2010. For the best trade-offs, clearly defined measurements for these criteria must be routinely evaluated.[59]

Consequently, the following tangible contributions are made in this chapter:

1. Machine usage, energy consumption, and service level agreement infractions are established or reused as formal models.
2. By building off of and expanding upon the Machine Reassignment model offered by Google for the EURO/ROADEF competition, we provide a tangible formulation of the Service Consolidation issue for a real-world cloud stack.

3. By simulating clouds of varying sizes, shapes, and workloads, a practical consolidation approach for SLA compliance is shown. By taking into account service and machine level limitations, meta heuristic search is used to uncover (re)allocation plans (solutions) that transition the cloud from a decentralized to

centralized state.

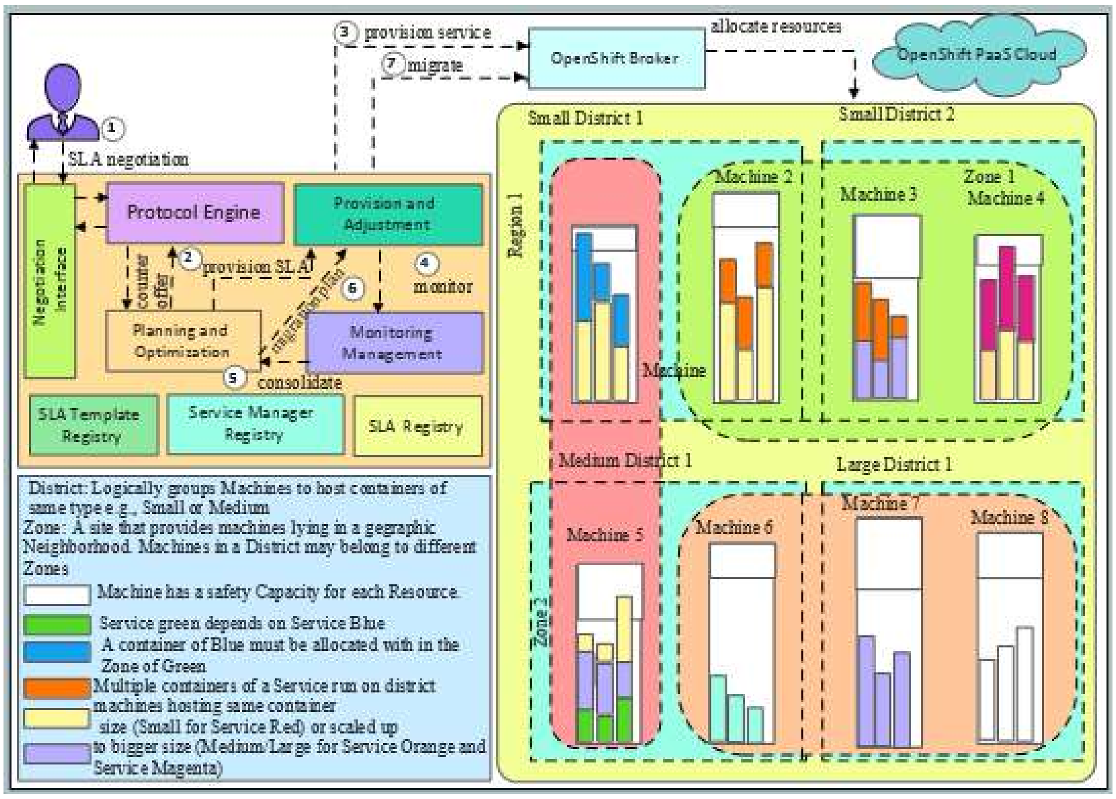
4. Analysis of the effectiveness of the solutions found using Tabu Search, Simulated Annealing, Late Acceptance, and Late Simulated Annealing when the problem's attributes shift across datasets.

5. Alternatives are ranked in accordance with different company principles using a utility function.

3.2.1 System Context

The SLA@SOI project's Generic SLA management framework (GSLAM) was outlined. To implement SLA management on cloud stacks, this may be created. OpenShift-based SLAM's condensed architecture is shown in Fig. The diagram displays significant interaction patterns:

1) The Protocol Engine is used to negotiate a Service Level Agreement (SLA) that specifies agreed-upon Quality of Service (QoS) values. When an offer is received, the POC module is used to analyze the data and come up with a counteroffer.
2) The POC initiates provisioning using the Provisioning and Adjustment subcomponent after a SLA has been satisfactorily agreed upon.
3) OpenShift's broker is called upon by provisioning to distribute resources.
4) Monitoring The administration initiates a system based on OpenTSDB to track resource use vs SLA.
5) If underutilization is detected over time, the monitoring system will submit a request for Service Consolidation to the POC.
6) To generate a migration strategy, POC optimizes the model of how allocated containers and machines are used.
7) The migration strategy is implemented through provisioning at non-busy hours.



District: Logically groups Machines to host containers of same type e.g., Small or Medium
Zone: A site that provides machines lying in a gegraphic Neighborhood. Machines in a District may belong to different Zones

⬜	Machine has a safety Capacity for each Resource
🟩	Service green depends on Service Blue
🟦	A container of Blue must be allocated with in the Zone of Green
🟧	Multiple containers of a Service run on district machines hosting same container
🟨	size (Small for Service Red) or scaled up
🟪	to bigger size (Medium/Large for Service Orange and Service Magenta)

Figure 3.9: OpenShift SLAM, OpenShift Cloud and Legend (elaborating container placement) (E.Yaqub et. al,2014)

The OpenShift cloud consists of one or more brokers that take API requests to manage services and cloud machines. Districts are collections of machines that may come from many IaaS suppliers. A zone is assigned to an IaaS site. There are several zones within an area. OpenShift cloud may span various domain infrastructures (zones) across different continents to increase resilience and permit compliance with local data protection requirements (regions). Custom algorithms for container placement can be included into its plugin-in interface. The POC of our SLAM may be added to this.[60]

3.2.2. Model and Definition of the Issue

The Machine Reassignment problem, which Google submitted for the ROADEF/EURO Challenge 2012, is what inspired this study because of how well it applies to PaaS. The datasets offered by Google are common ground for future exploitation and pose actual problems. Therefore, we essentially follow the specification, but we also take OpenShift into account for altering various restrictions and for dataset characterizations because without them, the problem would mostly stay theoretical. This strategy offers generally recognized definitions while avoiding workload bias.

The main goal of the issue is to increase machine usage given a set of services. A service consists of a collection of tasks that are given to machines and use their resources. Because a container is the smallest deployment unit in PaaS, we refer to processes as containers. To increase utilization, containers can be moved across computers; however transfers are restricted by both hard and soft limitations. Both concepts of machine apply since OpenShift PaaS may be constructed on physical or virtual computers. We then officially outline the issue and the models.

3.2.2.1. Notations

Consider In this notation, M represents a collection of machines, V represents a collection of virtual containers, and R represents a collection of shared resources accessible by all machines. An S is both a collection of services and an individual service $s \in S$ is a set of containers $\subset V$. Services are disjoint. $M(v) = m$ is an assignment of a container $v \in V$ to a machine $m \in M$ where v is the first container to be assigned the value Mo(v). Container v has a need for Resource R, therefore R(v, r) is the sum of the capacity of Resource r R for Machine m and the need for Resource r by Container v. The safety capacity of resource r for device m is denoted by the formula SC(m, r).[61]

Let T = {small, medium, large} include a collection of container sizes, with the little ones being supported such that < medium < large along all resource dimensions. Let D represent a collection of regions, whereby each region (d) $\in D$ represents a collection of machines. Municipalities are examples of sets that are not connected. In this notation, a machine's district is denoted by d(m), a container's type is denoted by t(v) T, and the container type of the district to which a machine belongs is denoted by t(d(m)). For instance, a district of type large can host large, medium, and small containers, a district of type medium can host medium and small containers, but a district of type small can only host small containers. Let there be a set Z of disconnected regions, where each region $z \in Z$ is a collection of machines in the same general area. An IaaS availability zone is denoted by a zone. In this work, zone is equated to the neighborhood (of Machine Reassignment problem).

3.2.2.2. Hard Constraints

To illustrate the inflexibility of hard limitations, consider the following example. One definition of a workable solution is one that fulfils all inflexible requirements.

Definition 1 : Capacity Constraint

The utilization (U) of a machine (m) with respect to a given resource (r) is defined as:

$$\mathrm{U}(m,r) = \sum_{v \in \mathcal{V}, M(v)=m} R(v,r)$$

Only if there is enough of each resource on the host computer is it possible to execute a container. When the capacity limitation is met, the assignment is considered practical.[62]

$$\forall m \in \mathcal{M}, r \in \mathcal{R}, \mathrm{U}(m,r) \leq C(m,r)$$

Definition 2 : Type Constraint

An appropriate container is then installed on a machine in a region where it may be used:

$\forall v \in V, M(v) = m, t(v) \leq t(d(m))$

This permits scaling upwards, but not downwards, since it would be a breach of the SLA's resource capacity promise.

Definition 3 : Conflict Constraint

Each computer hosting a container for a service s ∈ S must be completely separate from any others.

$\forall s \in S, (v_i \, v_j) \in s^2, v_i \neq v_j \Rightarrow M(v_i) \neq M(v_j)$

Definition 4: Transient Usage Constraint

Some resources, like as disc space or RAM, are used twice when a container is moved from one host computer to another. As a result, the migration requires sufficient resources on both machines m and m'.Let T R ⊆ R are scarce resources subject to temporary use restrictions, then:

$$\forall m \in M, r \in \mathcal{T}\,\mathcal{R}, \quad \sum_{\substack{v \in \mathcal{V}, such that \\ M_0(v)=m \vee M(v)=m}} \mathcal{R}(v,r) \le C(m,r)$$

Due to incompatibility with our Type constraint, we disregard the Spread restriction.

3.2.2.3. Soft Constraints

The costs of resource contention, collocation needs, and migrations are modelled as soft restrictions. Original data sets often include a definition of costs. The extent to which soft restrictions are met indicates the quality of a possible solution.[63]

Definition 5 : Load Cost

Let SC(m, r) be the safety capacity of a resource r ∈ R on a machine m ∈ M. The load cost is defined per resource and refers to the capacity used above SC.

$$loadCost(r) = \sum_{m \in M} \max(0, U(m,r) - SC(m,r))$$

Definition 6: Dependency Cost

Earlier in this article, I said that there may be connections between various services. There should be consideration for how dependencies could affect the availability and performance of a service when allocating resources. To elaborate a little, a service s^a dependent on service s^b necessitates that each s a container run in the vicinity of at least one s^b container. Lack of fulfillment causes dependence to grow. Cost varying.

$$\forall v^a s^a, \exists v^b \in s^b s.t. M(v^a) \in z1 \; and \; m(v^b) \in z2 \;\; and \; z1 = z2$$

$$if(z1 \neq z2) \rightarrowtail dependencyCost + +$$

Technically speaking, the OpenShift Application Programming Interface (API) enables service migration to a different zone (or region). Using its software-based networking instructions, communication between containers may be built up after migration.

Definition 7: Balance Cost

The presence of one resource may be meaningless in the absence of another. An availability ratio between two resources is targeted for assignment after a container migration. One way to express this is using a group of triples that are in balance, denoted by the set B N X R2 . For a triple b =< r1, r2, target >∈ B, the balanceCost is:

$$balanceCost = \sum_{m \in M} \max\big(0, target * A(m, r1) - A(m, r2)\big)\ where\ A(m, r) = C(m, r) - U(m, r)$$

Definition 8: Container Move Cost

This expense represents the challenges associated with moving certain containers, such as those with heavy dependencies, excessive size, a lack of replicas/clones, or administrative constraints. The total cost of moving containers is calculated as the sum of each individual container's migration costs, denoted by CMC(v).

$$containerMoveCost = \sum_{\substack{v \in V\ suchthat \\ M(v) \neq Mo(v)}} CMC(v)$$

Definition 9: Service Move Cost

The maximum allowable number of containers to be moved is determined by the cost of the service relocation. This maintains a steady equilibrium between the various service migrations.

$$serviceMoveCost = \max_{s \in \mathcal{S}}\big(|\{v \in S | m(v) \neq M_o\ (v)\}|\big)$$

Definition 10: Machine Move Cost

Let MMC (m_{source}, $m_{destination}$) be the total amount of money it would cost to move container v from machine msource to machine mdestination. This fee simulates the difficulty introduced by factors such as physical distance, network topology, individual machines, and other technological factors, and is thus unique to each machine combination. The total cost of a machine relocation is determined by adding the MMCs that apply.[64]

$$machineMoveCost = \sum_{\substack{v \in V\ such that \\ M(v) \neq M_0(v)}} M\ MC(M_0(v), M(v))$$

Definition 11: Objective Function

An objective function is defined as the sum of all costs, using weights from the original datasets, for each soft constraint.

$$totalCost = \sum_{r \in \mathcal{R}} weight_{loadCost(r)} * loadCost(r) +$$

$$\sum_{b \in B} weight_{balancedCost(b)} * \ balanceCost(b) +$$

$$weight_{containerMoveCost} * containerMoveCost +$$

$$weight_{serviceMoveCost} * serviceMoveCost +$$

$$weight_{machineMoveCos} * machineMoveCost + dependencyCost$$

To reduce this number, we need to look for assignments that are both practical and fair in terms of resource contention among the computers that are being utilized. In contrast, unhappiness with constraints is quantified as a negative number (called a score) that increases when better solutions are found.

3.3. OpenShift Characterization

The OpenShift (version 2.0) paradigm is used to place the ROADEF data sets p ∈ P given in the datasets was characterized as a container v ∈ V of type t ∈ T. Figure 1 depicts the system used for categorization. R(p, r) represents the need for resource r by process p, whereas minR(P, r) and maxR(P, r) represent the least and maximum needs for resource r across all processes. When a process's utilization of a given resource is compared to the global usage of that resource, the s, m, and l flags are incremented accordingly (lines 7-15). A big container is one in which a process consumes a lot of a given resource, whereas a medium container consumes a little less, and a tiny container consumes very little.

Next, each group of machines called location $l \in L$ was characterized as OpenShift district $d \in D$ which hosts containers of subsuming types. The classification method is shown in Fig. Here, dominance count, which is calculated for all resources of all computers belonging to each location, demonstrates how one site dominates another over a resource. Using these characteristics, we classify the most dominant areas as major districts, followed by medium and minor districts, in that order.

```
class CharacterizedProcessAsContainer
method Classify(P,R)
1: for each resource $r \in R$ do
2: $min_r \leftarrow minR(P,r)$; $max_r \leftarrow maxR(P,r)$
3: $portion_r \leftarrow (max_r - min_r) / 3$
4: end for
5: for each $p \in P$ do
6: $usage_p = (s, m, l) \in Z^3$
7: for each resource $r \in R$ do
8: if $R(p,r) \leq (min_r + portion_r)$
9:  $usage_p.s{+}{+}$
10: else if $R(p,r) \leq (min_r + 2 * portion_r)$ then
11: $usage_p.m{+}{+}$
12: else if $R(p,r) \leq max_r$ then
13: $usage_p.l{+}{+}$
14: end if
15: end for
16: if $usage_p.s \geq 1$ and $usage_p.m = 0$ and $usage_p.l = 0$ then
17: p.containerType ← small
18: else if $usage_p.m \geq 1$ and $usage_p.l = 0$ then
19: p.containerType ← medium
20: else if $usage_p.l \geq 1$ then
21: p.containerType ← large
22: end if
```

```
23:  end for
```

Container Characterizer

The fraction of big, medium, and small containers in the dataset that are characterized is given.

3.3.1. Utilization and Power Model

The ability to increase machine usage is still a benefit and a difficulty for cloud providers. Data centre often only use 10–20% of their server capacity, according to reports. Consolidating workloads lowers the peak-to-average utilization ratio by running more workload on fewer computers, which increases utilization. By switching idle machines into low-power mode or shutting them off, consolidation can lower energy expenses and the accompanying cooling costs because most machines use up to 70% of their peak utilization energy even when they are not in use.

The consumption of one or two resources, particularly the CPU, has been used to represent power usage. However, recent developments in processor technology have led to CPUs that consume less energy, but memory, disc, and network components are now a larger portion of the overall amount of power consumed.[65]

```
class characterizeLocationAsDistrict
    Let $v_s \subset \mathcal{V} : t(v_s) = small$ and $v_m \subset \mathcal{V} : t(v_m) = medium$ and $v_l \subset \mathcal{V} : t(v_l) = large$
method classify ($\mathcal{L}$ , M , R,V)
1:  smallDistrictCount =← min(1,L(| $\mathcal{L}$ | / | $\mathcal{v}$ |)*| $v_s$|L)
2:  mediumDistrictCount =← (l,L( | / | $\mathcal{v}$ |) * | $v_m$ |L)
3:  largeDistrictCount ← min(1,L(| $\mathcal{L}$ | / | $\mathcal{v}$ |)*| $v_l$|L)
4:  Map<resource,capacity > resources
5:  Map <location,resources>locationCapacity
6:   for each location $l \in \mathcal{L}$ do
7:   for each machine m ∈ :$M\ l(m) = l$ do
8:   for each resource r ∈  R do
 9:    resources[r].addOrUpdate(getResourceCapacity(m,r))
10:  end for
```

```
11: end for
12: locationCapacity[$l$] ← resources; resources.reset()
13: end for
14: for each location $l_{out} \in \mathcal{L}$ do
15: for each location $l_{in} \in \mathcal{L} : l_{in} \neq l_{out}$ do
16: for each resource $r \in R$ do
17: $r_{out}$ ← locationCapacity[$l_{out}$].resources[r].capacity
18: $r_{in}$ ← locationCapacity[$l_{in}$].resources[r].capacity
19: if $r_{out} > r_{in}$ then
20: $l_{out}$.dominanceCount++
21: end if
22: end for
23: end for
24: end for
25: locationsList ← locations.sortOnDominanceCountDescending()
26: largeCounter =0 ,mediumCounter =0, small.Counter =0
27: for each location l ∈ locationsList do
28:  if largeCounter < largeDistrictCount then
29: largeCounter++;l.districtType ← large
30: else if mediumCounter < mediumDistrictCount then
31: mediumCounter++;l.districtType ← medium
32: else if smallCounter < smallDistrictCount then
33: smallCounter++;l.districtType ← small
34: end if
35: end for
```

District Characterizer

According to research, the CPU used less than 30% of the power consumed by Google servers in 2007. Our datasets resemble point-in-time system snapshots rather than timeseries data, so we aggregate resource utilization to model machine utilization analytically:

$$u = (\sum_{r \in \mathcal{R}} \frac{U(m,r)}{C}(m,r)) \; / \; | \mathcal{R} |$$

where M is the overall capacity of the machine and R is the total capacity of resource r. The power model requires the usage of u.

$$P(u) = k * P_{max} + (1 - k) * P_{max} * u$$

where k represents the proportion of energy spent by the machine while it is not operating, and Pmax represents the highest amount of energy required when the machine is operating at full capacity. Our assumptions are that you're using a contemporary computer with a power consumption of 250 watts (Pmax) and a constant power input of k=0.7. The energy efficiency e of a cloud may be roughly calculated using the mean utilization umean of all the computers in use:

$$e = u_{mean}/P(u_{mean}) * 100$$

3.3.2. SLA Violation Model

Enhancing usage to reduce power costs runs the risk of over provisioning, which has a detrimental impact on performance. Controlling SLA breaches translates into this power-performance trade-goal off's of preserving earnings and reputation. We model SLA violations (SLAV) as upper bound estimates for the performance degradation caused by migration (PDM) and the performance degradation caused by contention for the machine's resources, while keeping in mind the state of the art. PDM is the period during which migrations cause containers to operate poorly or become unavailable. We assume that migrations are carried out in a serial fashion in order to assess the upper bound. Parallel migration undoubtedly lowers this number. As determined by PDM:

$$PDM = t_s * |v_s| + t_m * |v_m| + t_l * |v_l|$$

where vs, vm, and vl⊂ V are sets of small, medium, and big containers that might be migrated, and ts, tm, and tl are the maximum timeframes required to migrate a small, medium, or large container. Our studies use the following time constants: ts = 10s, tm = 20s, and tl = 40s.

When a resource is used beyond its safe capacity, the excess demand is recorded as load (r). Tertile 1 represents little disagreement, tertile 2 represents medium contention, and tertile 3 represents strong contention when measuring potential for dispute resolution

(PDC). A service level agreement (SLA) is broken only when resource utilization is more than 100% of total capacity and low contention (loadCost) is sought. The likelihood of service level agreement (SLA) breaches may be estimated using the tertiles to determine whether contention needs can be lowered by increasing safety capacity to tertiles 1, 2, or 3. This may increase utilization and reduce the MTTF of a machine at the cost of some extra electricity based on PDC's findings. [66]

$$Cont_{r \in R} = C(m,r) - SC(m,r)$$

$$PDC = \begin{cases} if\ load(r) \leq \frac{1}{3}(Cont_r) \mapsto tertile1++ \\ \quad else\ if \\ load(r) \leq \frac{2}{3}(Cont_r) \longmapsto tertile2++ \\ \quad else\ if \\ load(r) > \frac{2}{3}(Cont_r) \longmapsto tertile3++ \end{cases}$$

Machines' PDC tertiles are increased by one each time a new dataset is added. Finally, we define SLAV as the result of PDM and PDC, as follows:

$$SLAV = PDM * max(1, PDC)$$

In this context, PDC may be either (i) the total of all tertiles, (ii) the total of tertiles 2 and 3, or (iii) only tertile 3. This results in a range of SLAV values for the solved datasets of up to 3.

3.3.3 Results of Experiment

3.3.3.1. Workflow

Fig. depicts the steps that make up the experiment workflow. The problem and datasets are first placed within the context of the OpenShift PaaS cloud model. Next, a solver's algorithms are configured and the problem constraints and score calculation are implemented. The next step was to conduct a controlled search to identify the first workable solution. This suggested redundancy among larger containers because scaling down is not possible while scaling up is. As a result, the number of randomly chosen large containers—the majority of which are independent—was decreased. Due to the fact that OpenShift-based refinements make the search more difficult than the underlying issue, this helped introduce some slack.

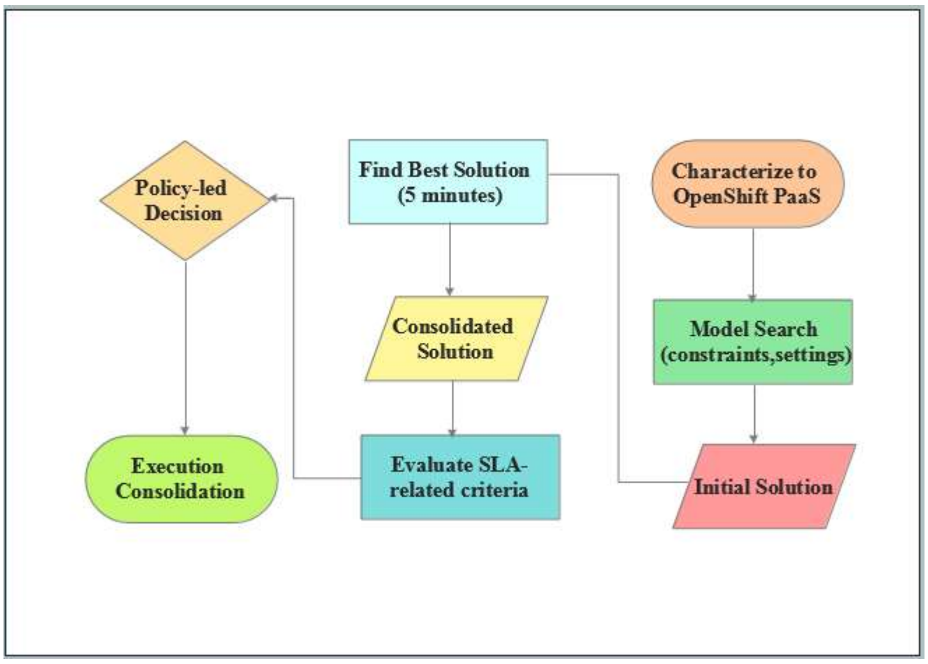

Policy-led Decision
Find Best Solution (5 minutes)
Characterize to OpenShift PaaS
Consolidated Solution
Model Search (constraints,settings)
Execution Consolidation
Evaluate SLA-related criteria
Initial Solution

Figure 3.10: Workflow of Experiment

The base definition does not limit machines to hosting containers of a specific size or categorize processes according to their resource needs, such as small, medium, or large. When tackling the issue of service consolidation from a real-world cloud perspective, bin packing becomes a formidable challenge. The next step is to initiate a search, giving each algorithm 5 minutes to complete its task. Ultimately, each method yields a unified (reallocation) answer. Utilization, resource contention, migrations, service level agreement breaches, machine use, and energy consumption are now among the additional assessment criteria assessed from proposed solutions. As a consequence, the optimal solution is identified and implemented via policy-led ranking of the available options. To improve performance, OpenShift's API allows containers to be moved around and their capacities increased during runtime.[67]

3.3.3.2. Datasets

Through the use of simulations, we were able to reliably assess the performance of various approaches on the data sets presented. Table 3.3 provides a description of the individual datasets that do not form a unified whole. In order to use it later for spotting purposes, we archive the dataset. A dataset is a representation of a workload and associated cloud configurations. Datasets A.2.2 and A.2.5, which both represent small clouds, have 100 and 50 machines, respectively, hosting 170 and 153 services. Dataset B.1 shows a cloud of a medium scale, with 100 machines running 15 times more services than smaller datasets like this one. Dataset B.4 is an example of a very big cloud, consisting of 500 individual computers, with more containers than in Dataset B.1 but fewer services.

A theoretical upper limit was determined for the state space by supposing that any container of any size may be placed on any machine in any district. More possible permutations exist since container sizes may be increased. The diagrams represent a subset of the whole state space called the base state space. The purpose of this exercise is to highlight the vastness of possible permutations as a yardstick by which to evaluate dissimilar data sets. The purpose of search algorithms is to find a subset of the whole state space that is manageable, which is much smaller yet cannot be practically established due to the combinatorial explosion.

3.4. Algorithms

The needed variety to mimic real-world dynamics is provided by machine heterogeneity and various resource usages. Meta-heuristic search methods are strongly motivated to be used for solving because of the huge state spaces, high dimensionality caused by various resources, and numerous restrictions. Additionally, meta-heuristics avoid the pitfall of becoming trapped in local optima. This study made use of the open-source OptaPlanner solver, which supports several Meta-heuristics. All algorithms were a mixture of change and swap moves. One entry in a tabu list of size 7 that was utilized for tabu search (TS) indicates a workable assignment for a single container. A portion of the many viable motions that TS generates is assessed in each stage. This evaluation size was established in 2000 for TS. A beginning temperature value is necessary for

Simulated Annealing (SA) to factor score difference. This was set to 0 for strict limitations and 400 for flexible ones. The algorithm first permits certain changes that are not improving, but as time progresses, it becomes more exclusive. The assessment size for SA was set at 3. Like SA, Late Acceptance (LA) requires fewer actions while outperforming some late phases. The assessment size for LA was set to 500 and the late size to 2000. In Late Simulated Annealing (LSA), which combines SA and LA, the score can be improved while still experiencing controlled random decrement. The evaluation size was set to 5 and the late size to 100 for LSA.

Table 3.3.: Cloud Dataset Details

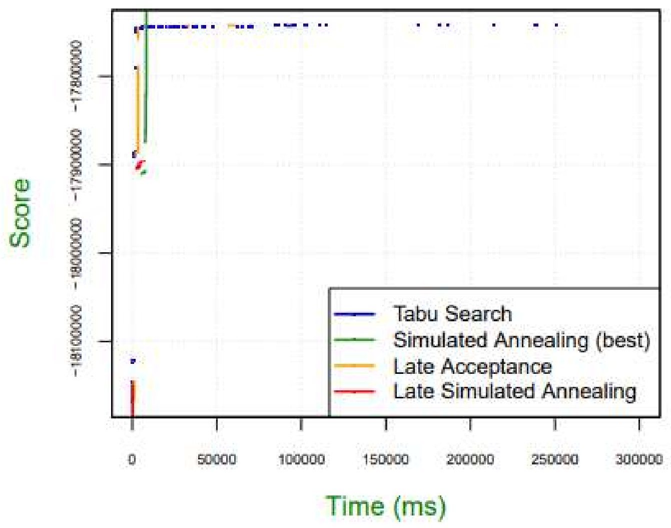
Table 3.3.: Cloud Dataset Details											
Name	**Resources**	**Districts S,M,L**	**Zones**	**Machines S,M,L**	**S32ervices**	**Containers S,M,L**	**State Space**	**Utilization/ Machine**	**Consumed Power(kWh)**	**Energy Efficiency**	**Initial Solution**
A.2.2	12	15,8,2	5	60,32,8	170	434,222,48	101148	35%	19.32	43.25%	-18312546
A.2.5	12	15,8,2	5	30,16,4	153	455,245,63	101004	37%	10.151	43.50%	-185103629
B.1	12	6,3,1	5	60,30,10	2512	25,721,272,147	106598	48%	21.122	56.75%	-945528368
B.4	6	35,13,2	5	350,130,20	1732	109,643,955,543	1036959	37%	101.199	45%	-18475614730

Table 3.4.: Consolidation using Tabu Search (TS), Simulated Annealing (SA), Late Acceptance (LA) and Late Simulated Annealing (LSA)

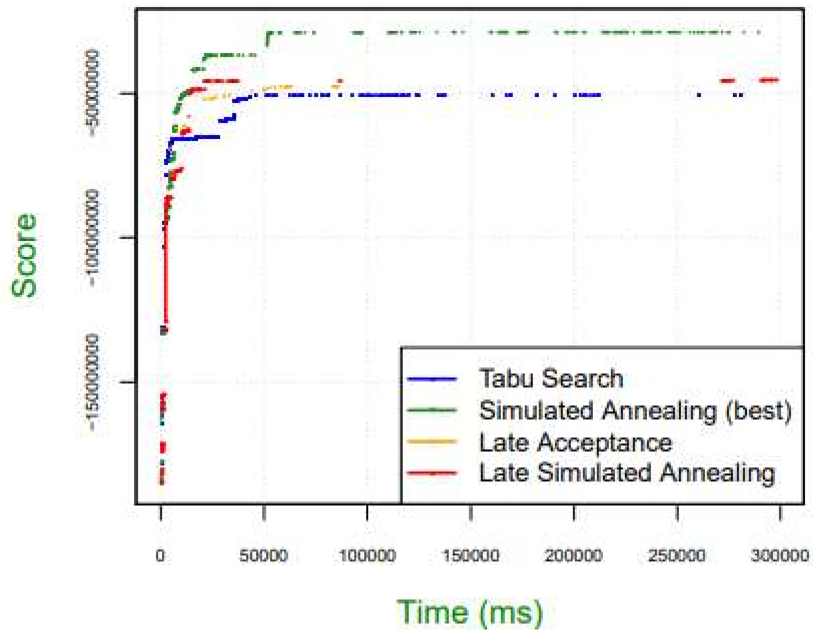
Dataset (Algorithm)	Solved SolutionScore	MachinesUsed S+M+L	Utilization/ Machine	Consumed Power(kWh)	PDCTertiles(1,2,3)	ReducedLoad onResources	Energy Efficiency	Migrations	PDM (min)	Container Scaleup
A.2.2(TS)	-17742166	95	35%	19.141	(1,1,1)	50%	43.43%	206	66	85%
A.2.2(SA)	-15012210	95	35%	19.126	(0,0,1)	83%	43.46%	220	69	80%
A.2.2(LA)	-17741965	95	35%	19.144	(1,1,1)	50%	43.42%	205	66	85%
A.2.2(LSA)	-15023979	95	35%	19.092	(0,0,1)	83%	43.54%	369	95	40%
A.2.5(TS)	-50622223	47	40%	9.618	(17,4,4)	58%	48.87%	345	119	34%
A.2.5(SA)	-28774923	46	41%	9.454	(3,6,8)	71%	49.87%	351	119	30%
A.2.5(LA)	-47511660	47	40%	9.621	(22,3,4)	51%	48.85%	352	120	34%
A.2.5(LSA)	-45311736	47	40%	9.619	(20,2,4)	56%	48.86%	461	136	21%
B.1(TS)	-642113741	99	49%	20.951	(77,10,6)	12%	57.89%	2237	531	9%
B.1(SA)	-751389259	98	49%	20.766	(64,14,4)	23%	57.81%	2362	551	10.50%
B.1 (LA)	-544596716	99	49%	20.948	(72,14,4)	15%	57.89%	2237	532	9%
B.1(LSA)	-691537717	99	49%	20.947	(73,15,6)	11%	57.90%	2935	647	7%
B.4(TS)	-17116530373	500	36%	101.122	(0,0,0)	100%	44.50%	14883	3384	17%
B.4(SA)	-17136833433	500	36%	101.173	(18,4,2)	93%	44.48%	15176	3433	14%
B.4 (LA)	-17116172349	500	36%	101.153	(0,0,0)	100%	44.49%	14598	3333	17%
B.4(LSA)	-17136527221	500	37%	101.21	(17,3,4)	93%	43.70%	15207	3437	14%

3.4.1.Performance Results and Discussion

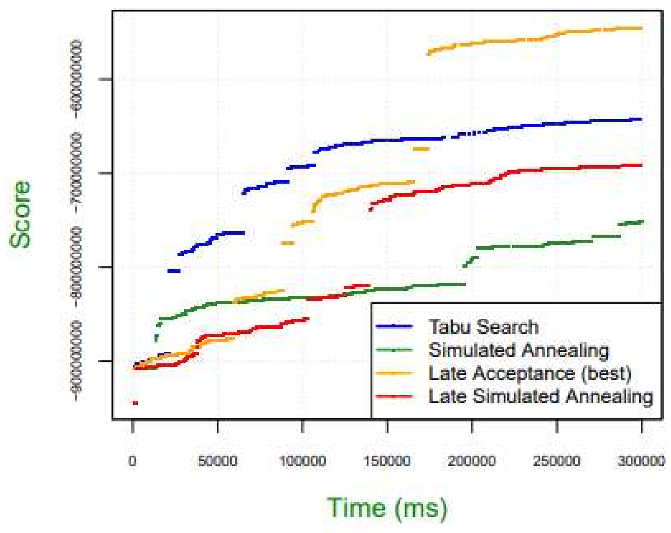
Table 3.4 lists the outcomes of the condensed solutions. The search pattern for score improvement is shown in Figs. 3.18(a-d). This makes it easier to assess how quickly a certain algorithm finds the best solution.



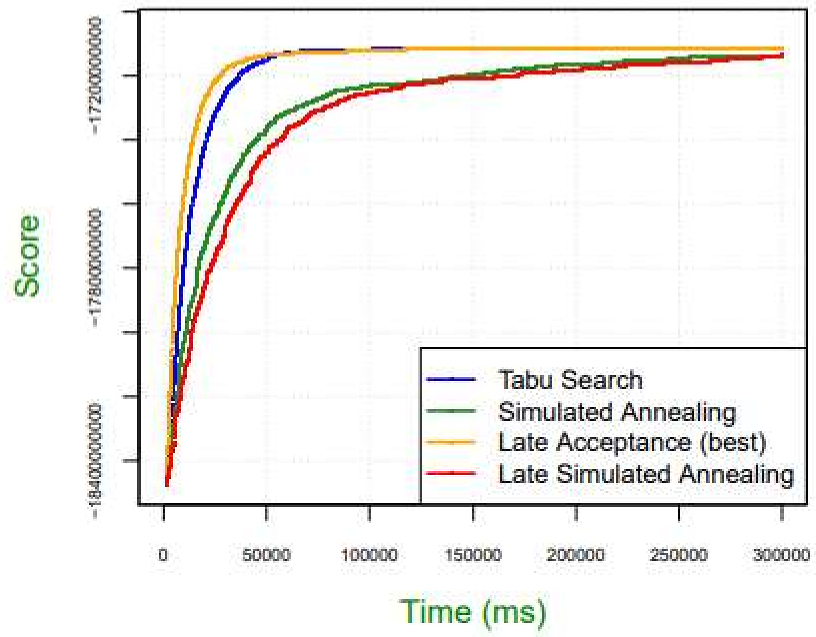
(a)



(b)



(c) Dataset B.1



(d) Dataset B.4

Figure 3.11.: Score Improvement Pattern of Algorithms(E.Yaqub et al.,2014)

The best score in (a) is found by SA in under 15 seconds, with LSA close behind. After 20 seconds in (b), SA shines brighter than LSA, which comes in second. Note that after just 50 seconds, the curve flattens out for SA, indicating that search has been exhausted and presumably the best solution has been located. As of yet, there hasn't been much work done to boost scores in other algorithms. When looking at criterion (c), it's clear that the pattern of score improvement is very irregular for LA and LSA, but not as irregular for TS and SA. After 175 seconds, LA has a commanding advantage against TS. However, no method exhibits a flattening of the curve, suggesting that the score might be improved even more in this dataset with further searching. Curves tend to flatten down in (d), revealing an intriguing pattern. However, for LA and TS, this flattening occurs quite quickly. After 100 seconds, LA is in the lead, and there is very little fight from TS after that.

Solved solution scores reveal that SA excels in the case of small state spaces, LA in the case of medium state spaces (B.1), and LA slightly outperforms TS in the case of large state spaces (B.4). Consolidation leads to a marked improvement in scores (as shown in Fig. 3.19). In a meta-analysis of all datasets, SA was found to have the highest score improvement (32.65%), followed by LSA (32.04%), LA (31.79%), and TS (28.19%).[68]

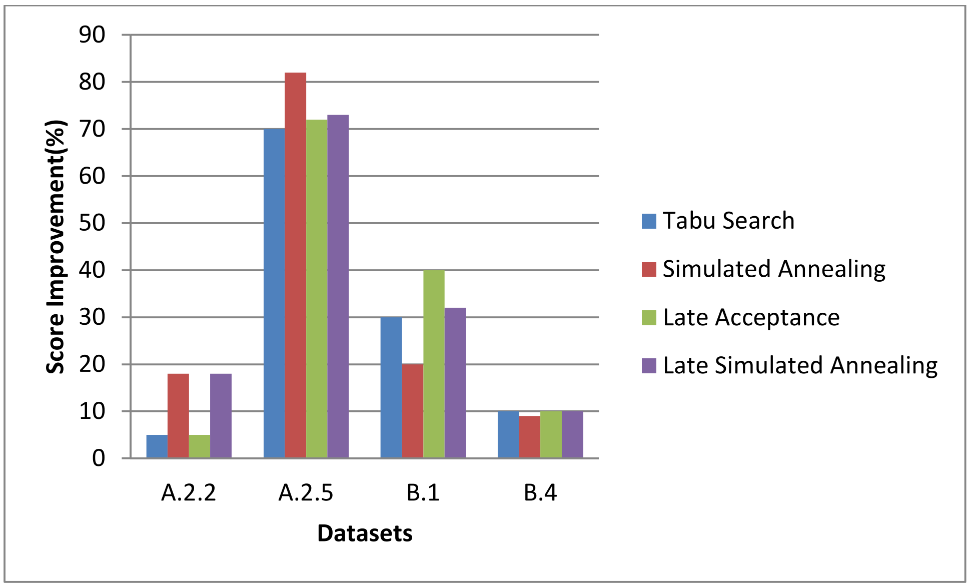
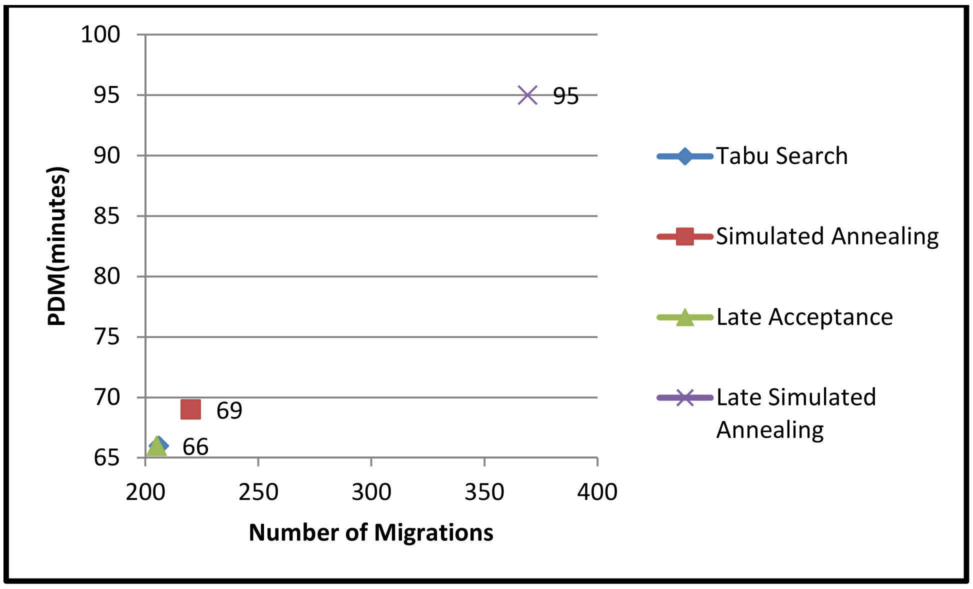
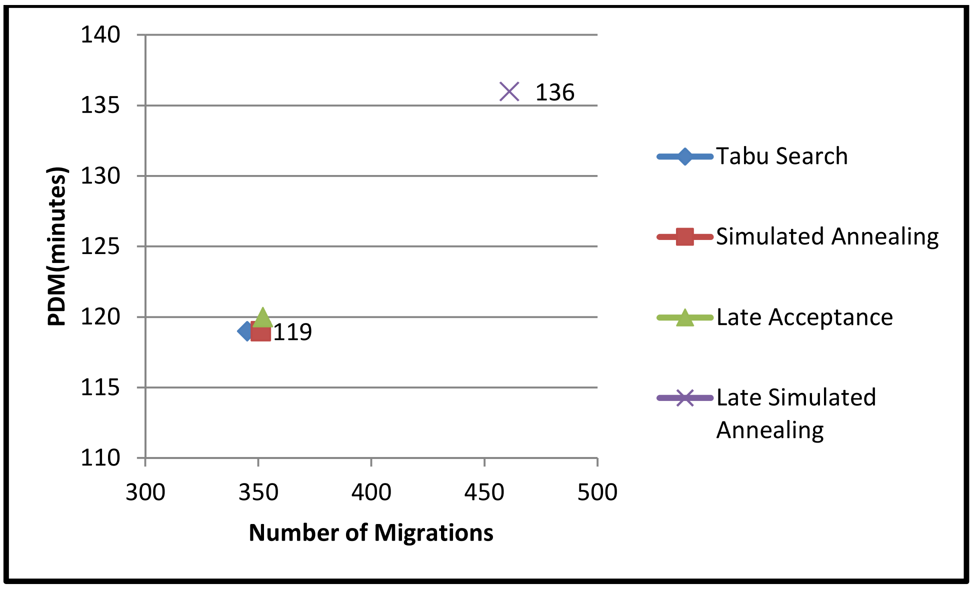


Figure 3.12.: Score improvement over initial solution

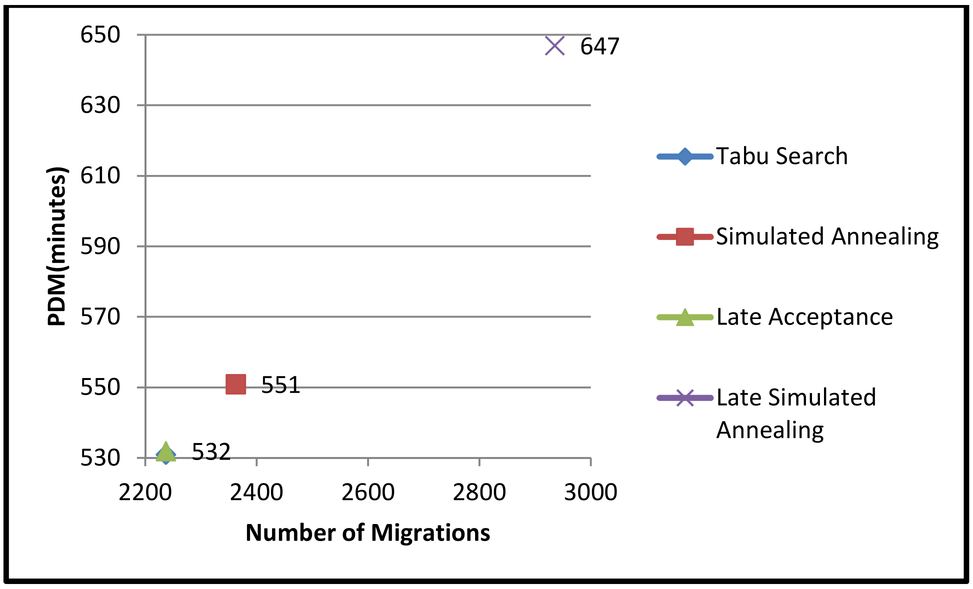
The number of container migrations (and associated PDM) suggested by the top method are shown in Figs. 3.20(a) through (d). SA, whose solution score was the highest, has a little disadvantage in plot (a) compared to TS and LA. Although TS once more suggests the fewest migrations in (b), the margin with SA is relatively small. SA is a definite winner given its large lead in solution score. In (c), LA and TS both offer the same number of migrations, but LA should be chosen since it outperforms TS in terms of solution score by a significant margin. In (d), LA suggests the fewest migrations and sets itself out from TS. The LSA makes the highest migration suggestions and is a blatant outlier. LSA is not a suitable option if tight minimization of migrations is desired.



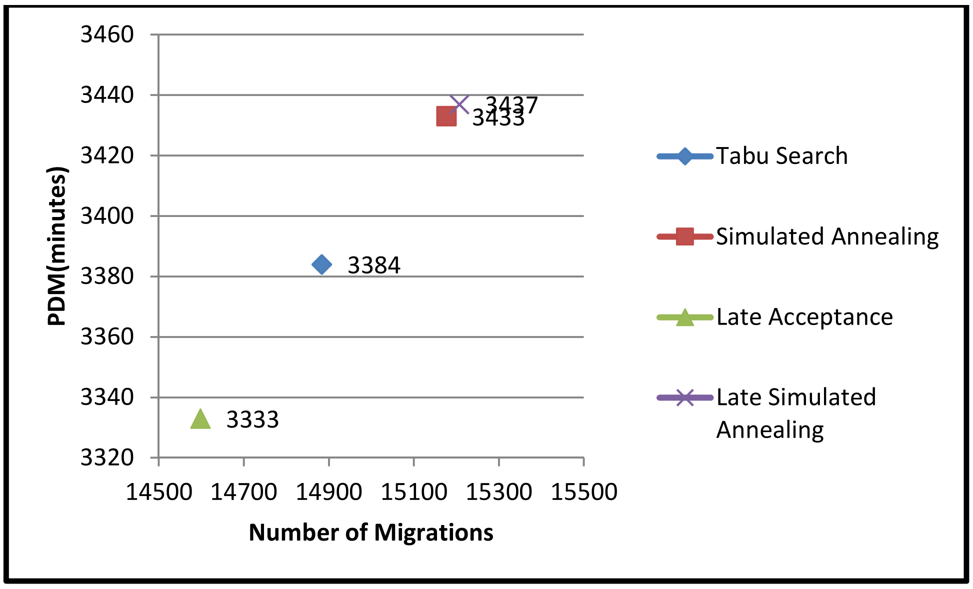
(a)



(b)



(c)

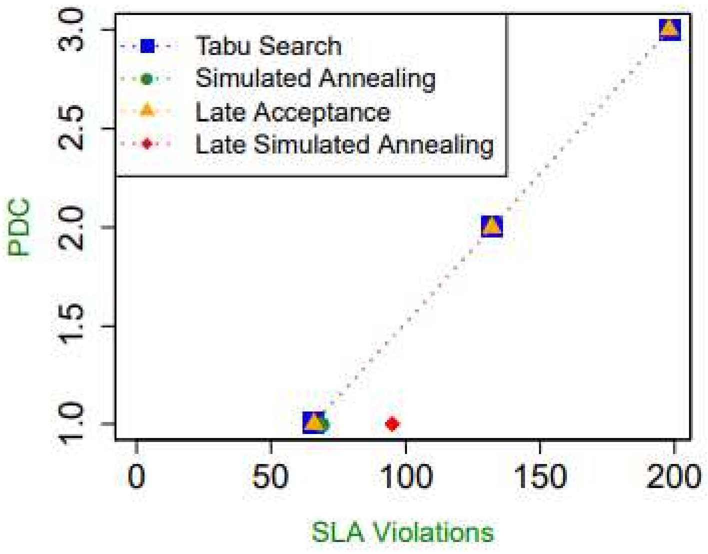


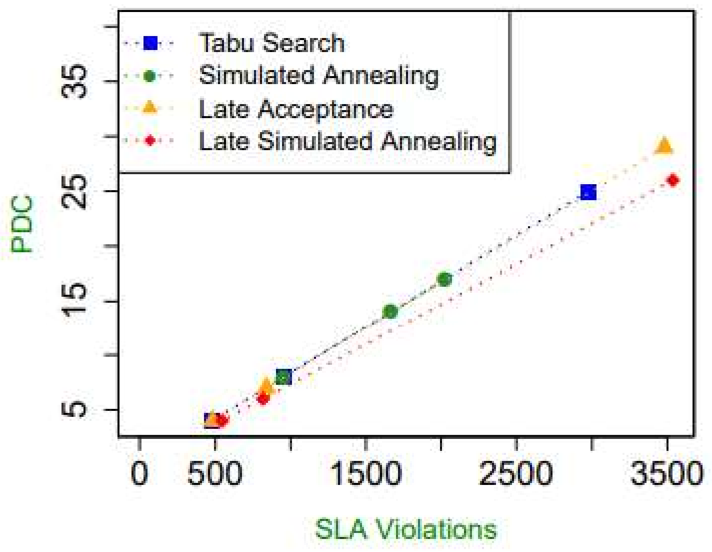
(d)

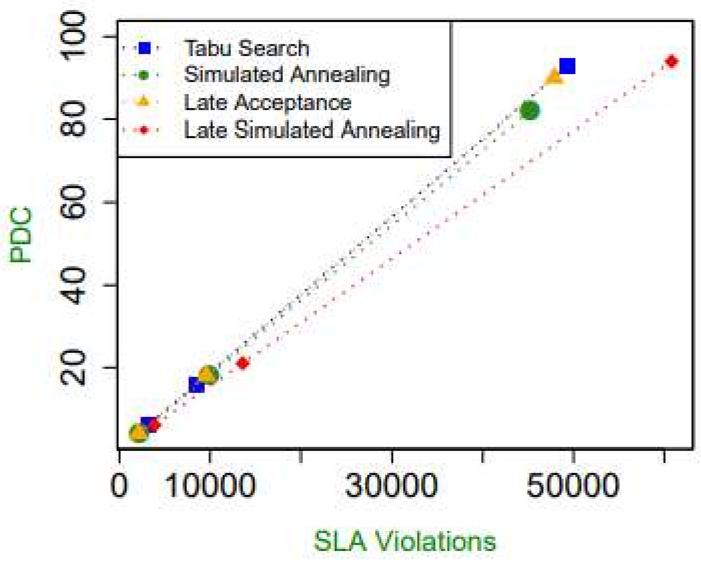
Figure 3.13.: Number of Migrations and PDM

Figs. SLA violations (SLAV) are displayed in 3.21(a–d) with lax PDC tertile values. Three distinct regions are plotted against the three PDC values in plot (a). As the strictest evaluation criterion, least contention (PDC=sum of all tertiles), SA and LSA both meet the cut in (a), but SA suggests a low SLAV because of its low PDM. In (b), SA defeats TS despite having a low SLAV because their PDMs are equal. In (c), LA comes in second place to SA and again produces low contention and, consequently, low SLAV. Amazingly, LA and TS eliminate all contention on 500 machines in (d).

LA gives least SLAV due to lower PDM than TS




Tabu Search
Simulated Annealing
Late Acceptance
Late Simulated Annealing
PDC
5
15
25
35
0
500
1500
2500
3500
SLA Violations


Tabu Search
Simulated Annealing
Late Acceptance
Late Simulated Annealing
PDC
20
40
60
80
100
0
10000
30000
50000
SLA Violations

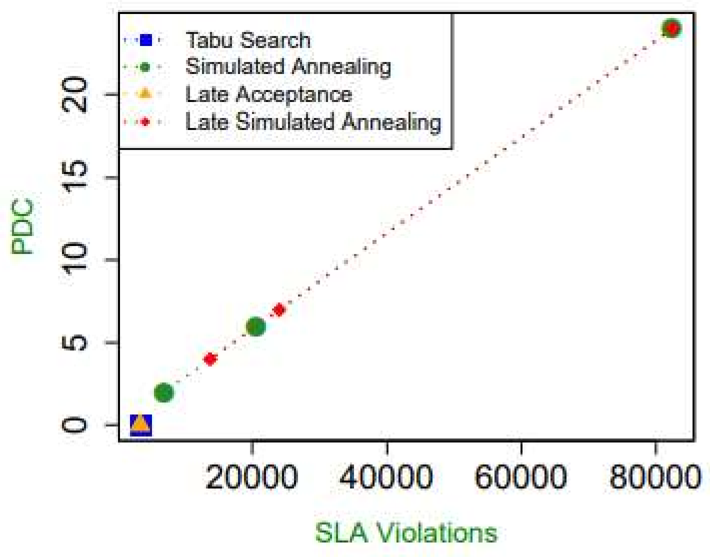


Figure 3.14.: Drop in SLA Violations with relaxing PDC as: i) sum of all Tertiles (north-east region), ii) sum of Tertile 2 and 3 (mid-region), iii) Tertile 3 (south-west region) (E.Yaqub et al.,2014)

When comparing the consolidated and unconsolidated system states using the same assessment criteria, Fig. 3.22 shows a dramatic drop in SLA breaches. The LSA is the lone outlier; it raised the SLAV in dataset B.1 by 0.02% when the PDC criteria were applied. However, when relaxed criteria are used, it excels. By averaging the findings from Fig. 3.22 over all datasets, we can evaluate the effectiveness of each approach in reducing SLA breaches. SA has the most decrease in SLA breaches, with a maximum of 70.38%; TS comes in second with 60.1%; LA and LSA tie for third with 59.43%; and LSA comes in last with 58.38%.

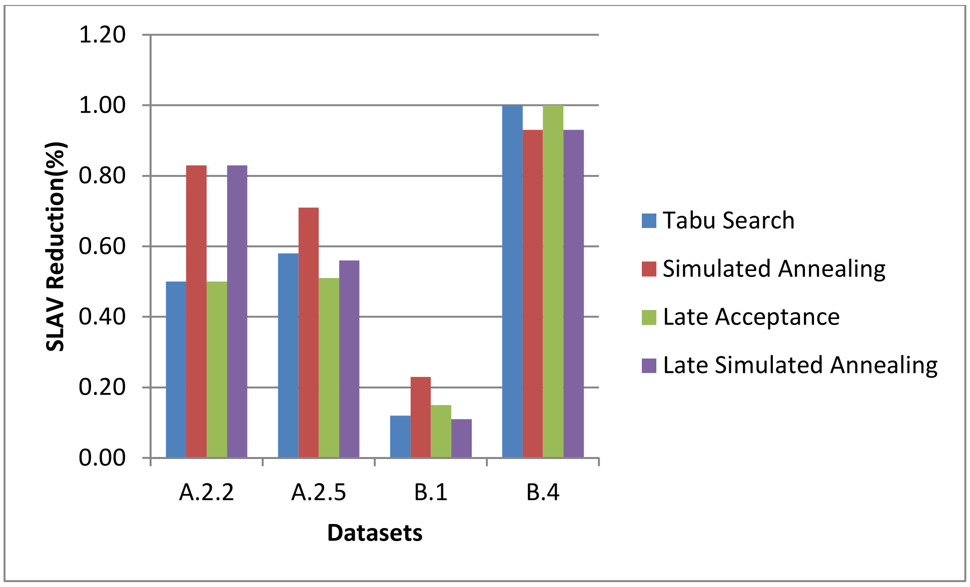


Figure 3.15.: Decrease in SLA Violations over initial solution

Overall, consolidation improved mean utilization of machines while using fewer machinery and, in most circumstances, less energy. A2.2 reduces resource burden by up to 83%, uses 1% less energy overall, keeps utilization same, but uses 5% fewer machines. A2.5 reduces resource load by up to 71%, saves energy by 7%, increases utilization by up to 3%, and employs 8% fewer machines. B.1 reduces resource burden by up to 23%, uses 2% less energy, increases utilization by 1%, and employs 2% fewer machines. The use of the same number of computers and a reduction in resource use of up to 100% in B.4 suggest that a lengthier search may be attempted on a wider scale. The energy consumption is regarded as the average figure for consumption per hour. Be aware that the 1-7% energy savings recorded here add up to respectable monthly savings, as illustrated in Fig. With monthly savings of 229.14kWh, SA is our most environmentally friendly option, followed by TS with 172.8kWh, LA with 166.68kWh, and LSA with 166.32kWh, according to the mean value across all datasets.

These findings support the idea that lower energy costs and fewer SLA breaches are the main benefits of service consolidation. This is accomplished by executing migrations that evenly distribute the workload among the computers, greatly reducing the load Cost therefore. Energy-efficiency and container scaleup numbers are also included in Table 3.4. The energy-efficiency of the utilized machinery must be taken into account. Values have been standardized as a result to get the highest energy efficiency with the employed machines.[69]

3.4.2. Policy-led Ranking of Solutions

Decision-making challenges for cloud service providers are analyzed and discussed. The author argues that the supplied metrics should be used and negotiated in line with an overarching policy. To this end, solutions are scored by average their usefulness across all metrics, with weights set according to the present policy's emphasis on various commercial considerations. The following utility function enables this:

$$U(sol) = \sum_{i=1}^{N} w_i u_i(x_i)$$

Here, U represents the value of a given solution sol, ui(xi) represents the value of metric xi as a real number between 0 and 1, and wi represents the weight placed on ui(xi). And $\sum_{i=1}^{N} w_i = 1$. This policy-led, utility-oriented scheme allows to obscure the complexity of individual metrics and decide for the most preferred solution.

Nine factors were standardized for use in assessing value to individuals. The solution score was normalised by the biggest difference between the original and solved scores. Equipment utilized and energy used were also comparable before and after consolidation. As the average utilization and load reduction numbers have already been normalized, they were used without further modification. There is evidence in some of the metrics of bad behavior. These consist of migration frequency, migration count, PDM scaleups, and the sum of PDC tertiles. To determine their positive utility, the discrepancies in their values were first normalized, and then they were subtracted from 1. Since energy-efficiency interpretation is prone to negative effects, it was not considered. Then, for every dataset, five different business policies are provided to rate algorithms and, by extension, solutions. The High Score Policy favours solutions with a high score and assigns it a weight of 0.5, while the other metrics are each assigned a weight of 0.0623. Those findings are shown in Table 3.5.

Table 3.5.: Algorithms Ranked on High Score Policy

	Datasets			
Rank	**A.2.2**	**A.2.5**	**B.1**	**B.4**
1	SA	SA	LA	TS
2	LSA	TS	TS	LA
3	TS	LSA	SA	SA
4	LA	LA	LSA	LSA

Low Migration Policy: The PDM and the number of migrations are each given a weight of 0.25 under this policy, which supports modest migration rates. The remaining metrics are each given a weight of 0.071428571. Table 3.6 displays the outcomes.

Table 3.6.: Algorithms Ranked on Low Migration Policy

	Datasets			
Rank	**A.2.2**	**A.2.5**	**B.1**	**B.4**
1	SA	SA	SA	TS
2	TS	TS	LA	LA
3	LA	LA	TS	SA
4	LSA	LSA	LSA	LSA

Weighing the PDC and decreased demand on resources at 0.25 each, this policy prioritises minimal resource contention. All other indicators are weighted equally at 0.071428571. The results are shown in Table 3.7.

Table 3.7.: Algorithms Ranked on Low Contention Policy

Datasets				
Rank	**A.2.2**	**A.2.5**	**B.1**	**B.4**
1	SA	SA	SA	TS
2	LSA	TS	LA	LA
3	TS	LSA	TS	SA
4	LA	LA	LSA	LSA

Assigning a weight of 0.25 to both PDM and PDC, the Low SLA Violations Policy encourages few service level agreement infractions. All other metrics each get a weight of 0.071428571. Table 3.8 displays the findings.

Table 3.8.: Algorithms Ranked on Low SLA Violations Policy

Datasets				
Rank	**A.2.2**	**A.2.5**	**B.1**	**B.4**
1	SA	SA	SA	TS
2	LSA	TS	LA	LA
3	TS	LA	TS	SA
4	LA	LSA	LSA	LSA

This environmentally friendly strategy priorities conserving energy by giving a weight of 0.5 to energy savings and a weight of 0.071428571 to all other measures. Table 3.9 displays the findings.

Table 3.9.: Algorithms Ranked on Low Energy Policy

Datasets				
Rank	**A.2.2**	**A.2.5**	**B.1**	**B.4**
1	LSA	SA	SA	TS
2	SA	TS	LA	LA
3	TS	LA	TS	SA
4	LA	LSA	LSA	LSA

After a coarse-grained analysis, it was shown that, across all five strategies, Simulated Annealing triumphed 13 times on small and medium-sized datasets (A2.2, A2.5, B.1). On the most competitive dataset (B.4), Tabu Search outperformed all other policies and finished on top with five victories. On the medium dataset (B.1), late acceptance rated highest solely for the high score policy, whereas on the small dataset (A2.2), late simulated annealing ranked best for the low energy policy. According to the available literature, simulated annealing is the best algorithm for enacting the vast majority of policies and increasing ROI (ROI).

In this chapter, we solved the problem of SLA-aware resource management in the cloud that service consolidation poses for OpenShift PaaS. According to the author's understanding, this is one of the first publications on the topic to employ several Metaheuristic algorithms and assess their effectiveness using formally specified models for several different features. Extending assessments to new metrics, limitations, and datasets is a possibility in the future. The work that has been presented makes a stronger argument for SLA management on contemporary cloud infrastructures.[70]

CHAPTER 4

4.0. Enforcement of SLA for Security of Cloud Network

4.1 System and Fault Models

Defects (inadequacies in the physical infrastructure), faults (inadequacies in certain functions), mistakes (incorrect system behavior), and failures (incorrect system performance) may all be assessed with the use of system and fault modeling (deviation of a specified behavior). By comparing the details of a system model with those of its flaws, we can more accurately estimate the severity of each flaw and determine the best way to address it [71] A system model in a testing scenario could have all the working parts and basic setups for the implementation in their optimal state. The fault model, on the other hand, finds potential failure sites throughout the whole process, from start to finish, revealing possible misbehaviors of a single node or a distributed computing cluster. This is the process of tracing a series of occurrences back to their originating causes, which is known as fault identification at the system level. By addressing our recommended system model, fault model scenarios covering both of our proposed architectures for enforcing cloud SLAs, and, by extension, it constructs a full security posture to correlate enforcement facets, this chapter gives some food for thought. This research takes it a step further by dissecting each main component of such systems and proposing fixes for the flaws and restrictions that were identified. We anticipate seeing these models introduced in the paper's subsequent chapter.

4.1.1.When Participants are Loss Averse

Our first proposed architecture is discussed below; it's intended for use in the scenario where all involved parties are loss-averse and are communicating in real-time.

4.1.1.1. System Model

A system model in distributed computing is the sum of the interactions between all the nodes, processors, processes, and their related components across all the different OS. Together, their communication paves the way for the completion of an undertaking, such as fulfilling a service request made by a client and started by a processor from either a trustworthy or an untrusted network. The system model also depicts several algorithms or protocols under which at least two interacting players agree on their interaction cycle, outlining the specifics of how their communication will be established, begun, task performance, and terminated after all tasks have been accomplished. The service reference design also addresses problems with fault tolerance and conflict resolution. Regardless of the distance between the nodes in question, this method makes it easier to examine the messages being sent and received and the predicted communication pattern and temporal model that arises whenever the nodes in question interact.

The system model also examines the proper operation of each component and their required actions in a perfect setting. When determining how long a process will take to execute and finish a job, it is important to consider whether the system is configured to do those activities using an asynchronous or synchronous communication paradigm. Additionally, the subsequent interactions will be evaluated using the same presumptions because the communication paradigm we have selected is synchronous. Real-time is necessary for synchronous systems to exchange data with the appropriate processors.

We consider a scenario in which three entities—the cloud service subscriber (CSS), the cloud service provider (CSP), and the end users—are involved.[72]

- **CSS**

A subscriber to a cloud service is a person or organization that uses the service to access computing resources like memory and hard drive space. CSS resource needs are tied to the anticipated application burden that would be necessary to meet CSS business goals. The CSS will offer something like digital payment in exchange for these computing resources (and other related services, such disaster recovery). An individual CSS's computational resource needs are set by the CSS's intended use or by the needs of the enterprise using the CSS. A is the collection of (business or application) parameters with desirable values that must be satisfied in order to implement a particular CSS. These criteria include, but are not limited to, availability and response times. We presume that a CSP will be able to map A onto a relevant R, the application's resource requirements. The CSS is assumed to have a function E that, given a collection of metrics (or sensors) S from the CSP, returns true if and only if E(S) satisfies A. The object assigned a CSS value is sometimes referred to as an International Classification of Standardization (ICSS) item. We make the assumption that there is a verification technique VICSS that can vouch for whether or not the item ICSS fulfils the specification ICSS.

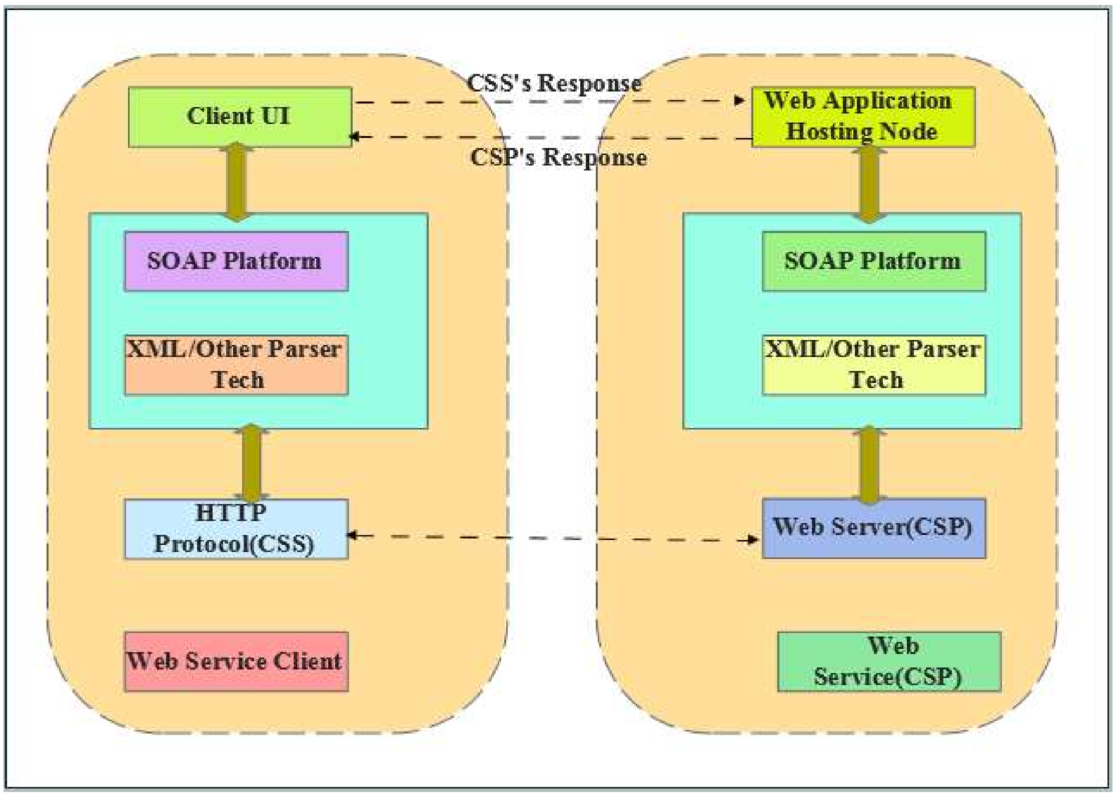


Figure 4.1: A basic Web Service Operating state

- **CSP**

At its most fundamental level, the CSP is responsible for supplying computing power to one or more CSSs. The CSP uses several different processes, including VM migration, scheduling, and elastic resource provisioning, to meet the demands of any

CSS that uses its services. Accurately estimating CSP adoption is outside the scope of this study. (For instance, given a CSS and an application requirement (A), a CSP must evaluate its available resources and any supplemental techniques to decide whether or not R resources may be devoted to A. (e.g., see). Let's say the CSS relies on the CSP to keep an eye on a set of signals (or sensors) S that, when taken together, reveal how successful the CSP has been in attaining A.A CSP may not be able to provide that all CSS criteria will be satisfied at all times because of the inevitable conflicts that will arise when serving several CSSs with different priorities. This is reflected in the service-level agreement (SLA) between the CSP and the CSS, which specifies the performance guarantees offered by the CSP. Each A application parameter will have an accompanying clause announced in the SLA. Each ensuing phrase is a (parameter) predicate. If E determines that any such criterion has been breached, the SLA stipulates that the CSS may escalate corrective action to the CSP. The CSP is assumed to periodically notify the CSS of S's value for assessment.

4.2.Message Security: Arrangements & Schemes:

In addition to examining the system model, it is recommended that the enabling technologies that protect (e.g., data security and privacy) message exchange channels and serve as required security implementations be assessed and studied. This underscores the need of parties assessing the security measures and procedures their service providers have in place to ensure the safe delivery of services to their customers' platforms.

4.2.1.Cryptographic Primitives:

When sending or receiving sensitive information through an untrusted channel, cryptography is a crucial component. Data security and privacy can never be ensured without the use of cryptographic techniques. In addition, security features like privacy, authentication, and credibility cannot be denied because of the use of these methods. Since we have used this approach in our work, it is necessary to explain how various cryptographic algorithms encrypt and decode messages in the thesis while assuming message exchange or exchanges between parties. Encryption occurs when a rendering process is applied to some intended data, rendering the data useless (i.e., illegible to anybody other than the intended users/recipients). These methods make it such that no one but the intended recipient may read any transmitted or received data (data at rest, data in transit). While an adversary can have access to encrypted material, they will not be able to read it without the correct cryptographic keys. Until an attacker can reverse-engineer the encryption rules and cryptosystems, the communication undergoes a transformation from plaintext to cipher text, in which the order of the message bits is jumbled. We'll have a look at a few methods for learning about the communication protocol and security assumption that's made while moving forward with the inter-communications arrangements of the participants. In Table 4.1, we can see several different schemes, each with its own set of characteristics and goals.[73]

Table 4.1: Cryptographic Schemes used in message exchange

Cryptographic Schemes			
Encryption Scheme	**Objective**	**Input**	**Output**
Symmetric Cryptography	Confidentiality	Plaintext(any length)	Cipher text(same length)
Asymmetric	Authentication	Plaintext(any length)	Fixed length signature
Asymmetric	Key agreement	Counterparty information	A session Key
Hash	Fingerprint	Plaintext(any length)	Fixed length message dependent fingerprint
pseudo-random number generators(PRNG)	Randomsness	Various	Hard to predict bits

4.2.2. Symmetric Encryption:

Secret key encryption, also known as symmetric key encryption, involves both the sender and the recipient keeping the same secret key in order to encrypt and decode the communication. Because the key is needed for both encryption and decryption, it is essential that both the sender and the recipient have a copy of the key. The message's privacy and integrity may be compromised if the secret key, which must be shared with the receiving person, were to fall into the wrong hands.

Symmetric users will be acquainted with a wide variety of encryption methods, including the Tiny Encryption Algorithm (TEA), Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), RC4, and Advance Encryption Standard (AES).

4.2.3. Asymmetric Cryptographic Algorithms:

In this method, also known as public key encryption, a user (A) encrypts a message (B) for another user (B), using B's public key (which may be known to others, such as unauthorized users), and then sends the encrypted message (B) to B for decryption. If the communication is intercepted by a third party, that third party will not be able to read it since they do not have access to User B's private key (which is only known to User B and no one else).Because of the nature of the mathematical process used as the foundation for one-way functionality, it is impossible to recover the original input values once they have been used to calculate a value. Everyone in this household is familiar with the cryptographic algorithms Diffie-Hellman and RSA.

Our study will be focused on the RSA method since it has been shown to be the most suited algorithm for usage in cloud computing environments and because it has been the focus of several studies and researchers.

4.2.4.Hybrid cryptographic Methods:

To guarantee the participants' integrity, confidentiality, and authenticity during their online interactions, a new trend involves the use of Hybrid Cryptographic Algorithms. This comprises the development of a hybrid cryptography combo with two distinct symmetric algorithms using simple integer variables and extended linear block cipher, and the creation of a combination of a symmetric key method of AES and the asymmetric key algorithm.[74]

4.2.4.1.Digital Signatures:

A digital signature guarantees the authenticity of a message by using a hash value that has been signed by the sender using their private key. Security-focused digital signatures have been widely adopted by businesses for use on a range of digital products. Digitally signing important documents or transactions conducted online increases confidence that the sender and recipient will be able to rely on the authenticity and integrity of the message. As messages travel back and forth over potentially unsecure channels, the use of digital signatures becomes increasingly important in establishing the reliability of the data being transferred. However, even if an attacker were to try to make some arbitrary changes to the data, such as by double spending, the digital signature would make that extremely difficult.

4.2.4.2. Non-cryptographic Methods:

In addition to cryptographic algorithms, other significant security measures are implemented to ensure channel security, such as traffic padding, routing control, passwords, smart cards, protected channels, and then other obvious security arrangements, such as a firewall.

4.2.4.3. Nonce:

In order to prove that the message being sent is not a repeating one, a "once" or "once-only" integer value is added in the message sequence. With nonce, an attacker cannot reuse a previously authenticated communication in a replay attack.

4.2.4.4. Digest Functions:

often denoted by the letter H, are safe hash functions. When going back over a conversation where messages were exchanged, it's important to note that H(M)=H(M') and that the messages should be viewed as having been modified as a result.

- **RSA**

RSA (Rivest, Shamir, and Adleman) has without a doubt established itself as the standard cryptographic method for creating public-key (asymmetric) encryption in recent years. It not only satisfies the protocol's authentication needs, but also executes its essential encryption features. The procedures involved in generating a key, encrypting the data at hand, and then decrypting it are outlined below:

1. To send a message to recipient B, sender A first produces the public-private key pair KpubA and KprivA. KpubA, the public key, is disseminated to an optimally safe place.
2. Here, A uses an authorized secure hash function to generate a message digest, H(M), and then encrypts the digest with his private key, KprivA, to generate a message signature, S=H(M)KprivA.
3. The sender (A) uses unreliable channels to send the intended encrypted message ([M]K=M, S) to the recipient (B).
4. If (B) decrypts S using KpubA and calculates the message digest of M, H(M), and they are the same, then (A)'s signature is valid.

Although RSA has been around for a while and has a good reputation as an encryption algorithm, it does have a few security flaws, the most serious of which is the possibility that an attacker with access to the private key could decrypt the entire data set. Only algorithms like Diffe-Hellman Encryption (DHE) that have the forward secrecy feature can protect against this danger, but DHE has its flaws, including a lack of speed. Ephemeral Elliptic Curve DHE (ECDHE-RSA) is a strong and quick asymmetric encryption method found in asymmetric encryption catalog.

4.2.5. Digital Certificates & Certificate Authorities (CAs)

With digital certificates, you know you're talking to the real deal while exchanging information with another party. If a distant user's digital certificate contains their public key and verifies that they've adhered to specific best practices or worldwide standards, it will deter attackers and unauthorized users from posing as that user. Certificates issued according to the X.509 (International Telecommunications Union) standard can contain information such as a serial number, issuer's name, validity term, subject's name, and public key, all of which verify the identity of the subject. Certificate authority are responsible for issuing certificates (CA). To ensure that only trusted CAs issue digital certificates, these companies provide digital notary services to anyone who are interested.[75]

4.2.5.1. Fault Model

In order to avoid a situation in which many nodes, processors, processes, connections, or service components of a system fail for various reasons, fault models must be created. Inaccurate processing times, (system, process, verification) values, and fabricated specifications can all be calculated and presented as a result of such mistakes. In addition to causing service interruptions, errors of this nature might compromise the security of the system as a whole, allowing unauthorized users access to the system or demonstrating that one of the serving nodes is acting arbitrarily. Omission (process/communication channel) failures, random byzantine failures, and timing failures, in addition to those categorized as benign, intermittent, and transitory faults, are all fully described in further study.

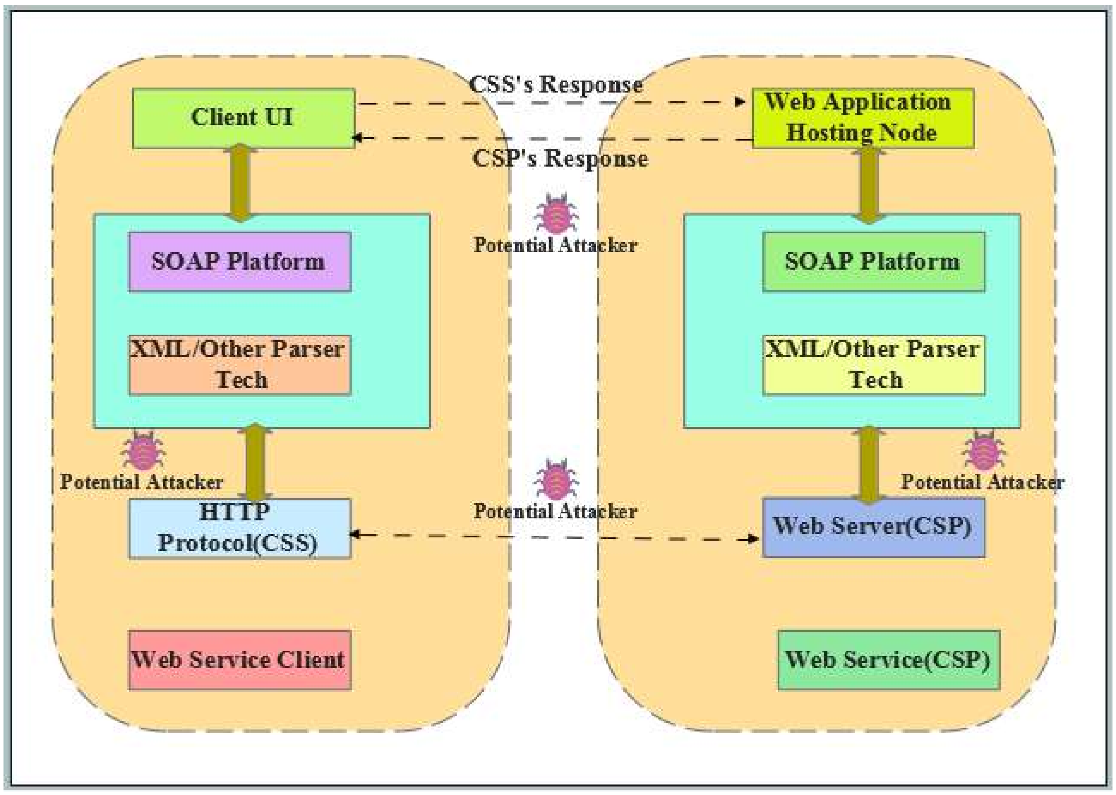


Figure 4.2: Potential attackers could cause failure to a web service operation

Web service security measures are implemented with a focus on the underlying web technologies via which web services are published, accessed, and even attacked:

a) Message Security
b) Infrastructure Security
c) Correct Response Generation
d) Authentic Recipient Only
e) Interaction Complies the SLA
f) Repel Attackers (internal/external)

If a CSP's supplied service element fails in an IaaS, PaaS, or SaaS environment, a SLA violation alert will be generated.

Our study focuses on faults that cause E(S) to violate A, therefore when we talk about a fault model, we're referring to those that prevent a CSP from meeting its SLA on the CSS. This might be the result of a malicious assault (i.e., external abuse) such as a distributed denial of service (DDoS) attack, a failure of the CSP's supporting procedures (i.e., design faults), or incorrect resource provisioning (i.e., component faults). The symptoms, in the form of Service Level Agreement (SLA) infractions, are of particular importance here.

In addition, we make the assumption that both the CSS and CSP are risk averse. A loss-averse participant (CSS or CSP) (i) will not attempt to sabotage the protocol's execution, (ii) will always accurately assess E(S), i.e., will not falsely accuse the CSP of breaking the SLA, and (iii) will always agree to the corrective action indicated in the SLA. This is because (i) for a CSP, a bad reputation may have a significant influence on the company's business strategy, and (ii) for a CSS, false accusations could result in lower availability while the violation is investigated and addressed, both of which are problems for a loss-averse player.[76]

When Participants are Malicious

While our system and fault model in the aforementioned examples only included three parties—the CSS, the CSP, and the end user—the number of parties involved in this interaction is growing as we expand our protocol's operational assumptions. Now we look at the other players; some of them are well-connected since they interact directly with one another, while others play a supporting role for primary players like CSPs and CSSs. Trusted Third Parties (TTPs) and Trusted Organizations (TOs) are examples of these broader players (TAs). For the organizations that have contracted with them, these reliable third parties serve as guarantors. When TACSS or TACSP receive an inquiry or request to provide materials committed by CSS or CSP, they are obligated and given a particular mandate to communicate with TTP. Since we want to account for these new players in our system model and fault model, we will need to make some adjustments to their dimensions and the scope of the models. While it's possible that many of the aforementioned operational limitations will remain the same, the scope of the fault model will inevitably be expanded to account for evident changes to our architecture and other related circumstances.

4.2.5.2. Threat Landscape:

When a web service in production falls victim to a cyber attack, it is a breach of the service level agreement (SLA). It might be an internal or external attacker, or something (such a changed procedure or a security breach) that puts the whole service at risk, or just a part of it. Errors within the scope of our study can happen anywhere between the CSP's trust boundary and the CSS's trust boundary, or on any of the communication channels between the two.

4.2.5.3.Cloud Attack Vectors:

There has been a dramatic increase in the adoption of cloud computing across all industries and regions of the world. In the wake of the current pandemic, cloud-based technologies have emerged as the frontrunners, giving decision-makers a low-maintenance service platform that nonetheless excels in reliability, safety, and regulation compliance. Since entering the office has become more difficult, businesses around the world have turned to cloud computing for all of their supply chain management needs. This includes businesses in the fields of international trade, education, shipping, healthcare, financial technologies (fin. tech.), food processing, retail, and wholesale. Revenues from cloud data centers, which store data and applications for businesses, are projected to surge to $304.9 billion by 2021, as demand for such services continues to rise across industries. The majority of these businesses will likely provide SaaS-based services.

This illustrates the increasing reliance of businesses throughout the world on cloud computing, a trend that, regrettably, also draws malicious actors. By utilizing a wide variety of attack vectors and tools, their end goal is the same: to shut down the affected cloud services. Secure, private, and readily accessible Critical data (wired or residual), edge computing infrastructure (physical or virtual), and endpoints are all targets of attack vectors. This poses a persistent threat to the safety of all parties involved in the cloud security industry, including service providers and their customers. If one of a company's vital services is hacked, the resulting lack of confidence in the security of the rest of the firm might be fatal. The attack vectors show their powers to the point where they may, in the worst case, disrupt cloud services. When unidentified assailants target vital services like healthcare or air travel, innocent people might lose their lives. When service outages are fought with these tactics, the subscriber suffers twice as much since they cannot substantiate their claims and must comply with SLAs or do additional service monitoring tasks.[77]

Web assaults, application attacks, and infrastructure attacks are only some of the targets of hostile actors' efforts. Attacks against SaaS clouds, PaaS clouds, and IaaS platforms may all be distinguished from one another, as can attacks on other service models.

When deploying a cloud service, it's important to think about how to secure the data services and who is responsible for them. Before agreeing to the SLA, further potential entry points for threats would be investigated as needed. Client and endpoint security, service access restrictions, hosting platform security, and identifying and confirming responsibility allocation are all part of this. Here are a few examples of those assaults:

a) Denial of Service (DoS) Attacks
b) Distributed Denial of Service (DDoS) Attacks
c) Cloud Malware-Injection Attacks
d) Cloud Side Channel Attacks
e) Authentication Attacks

f) Man-in-the-Middle Attacks
g) Trust & Reputation System At- tack
h) Cloud Service Containers (DDoS) Attack
i) Metadata Spoofing Attacks (WSDL service modification)
j) Server-side Request Forgery
k) Credential Stuffing Attack
l) Fake Cloud Services

Different service aspects, such as those in the technical, operational, or security spheres, are discussed here, along with their respective models and fault models. In addition to outlining these risks, the chapter discusses some methods for minimizing them during service exchange involving several parties. We foresaw how these business people's loss aversion would complicate matters when dealing with disputes. This chapter also covers situations when one or both participants could be up to no good, employing a wide variety of cloud-based assaults. Our following chapter details the operation of our suggested conceptual paradigm, including how it begins communication in the presence of reliable outsiders.

From the very first communication sent to the very last one received, this chapter covers it all. It describes the mechanics of implementing fully automated SLA enforcement.[78]

4.3. SLA Enforcement with Loss Averse Participants

The economic behavior known as "loss aversion" occurs when a company would rather operate at a loss than risk earning money that may be used to settle a lawsuit or repair its reputation if it were to go public. Their worry about financial loss is more important to them than the possibility of future financial gain. This business strategy tempts a company into momentarily deciding in favour of prospective unfavourable results over positive ones. Similarly, we see a first case where both of our primary business partners are averse to financial loss. Those involved in the sale of digital products for use in cloud settings, whether that be the service provider or a potential consumer, would do well to avoid any appearance of dishonesty. In addition, the cloud service provider will always be trustworthy when providing cloud services to the cloud service subscriber, and will never do anything to deceive the CSS, including tampering with the CSS's data security or privacy. At the other end, CSS would not exhibit any behavior indicative of malice, such as failing to pay the service provider or tampering with any product or service security measures.

This study first identifies the issue and the current limitations in cloud service delivery. In the first part of the paper, we covered the settings in which system models function, how they react when possible faults are injected, and the related system and fault models. That's why the study came up with a solution to the problem: a new kind of architecture. We posited that this is an issue with SLA enforcement, which is prevalent in the fair exchange. In the past, much of the effort went into tracking service level agreement (SLA) infractions. When it comes to monitoring, detecting, and enforcing cloud SLAs automatically, we know that an intelligent solution is a vital business necessity. To the best of our knowledge, no one has thought to use the fair exchange protocol in this way before. The theoretical implications of the suggested architecture were explained, and a micro-level implementation based on concurrent cloud technologies was supplied so that the design could be tested and findings could be gleaned from the experiment.[79]

4.3.1.Proposed Methodology

A rising number of cloud service providers (CSPs) are delivering a vast array of cloud-enabled services as the economic and technical advantages of shifting to a cloud-based computing paradigm become more evident. More and more CSSs are turning to CSPs for service guarantees regarding the quality of service (QoS) they can rely on to reinforce their business cases, drawn by capabilities like disaster recovery and variable resource availability.

A service level agreement (SLA) is a common tool for gauging a provider's quality of service (QoS) (SLA). Guaranteeing the CC and providing a guide for maintenance were its first roles. Different provisions in these SLAs address issues such as: I data ownership (e.g., Access to the data - data retrievable from CSP in readable format), (ii) the specific parameters and minimum service (e.g., Availability (e.g., 99.99% (peak), 99.9% for (off-peak) times or Performance (e.g., maximum response times), and (iii) system architecture and security levels or standards (e.g., Security). So, the SLA documents the CSP's guarantees to the CSS, who are leasing the services for payment.

Yet the system architecture that underpins these CC services is vulnerable to I failures, such as nodes crashing or being taken down or (ii) problems owing to targeted cyber assaults. The monetary cost to enterprises can be substantial under these conditions, and the CSP typically fails to meet its SLA obligations. When a SLA violation occurs, the clause dealing with such a situation kicks in. At some point, the CSS, as the harmed party, may try to get its money back using the SLA's provisions for business reimbursements. The time it takes to settle a disagreement through mediation is uncertain if it will be conducted manually. Strangely, despite being unable to provide proof due to lack of access to the CSP's computer estate, a CSS can be held accountable for a failed cloud service. The CSS must eventually follow a set of clearly established protocols, such as meticulously keeping track of all pertinent data and checking that it is both admissible and completely undamaged (digital forensics sound).

Moving toward automating the dispute resolution process will relieve the CSP of the burden of proof (with all the processes that must be adhered to). This third party monitors the situation to make sure neither party is treated unjustly. In this study, we present a cutting-edge approach based on the principle of equitable trade.[80]

4.3.1.1.Architecture Objective

In this part, we provide a high-level description of the issue we're trying to solve and then list the characteristics that any proposed solution must have.

Most of the components of the cloud service fabric adhere to the CSP-defined service-oriented architecture. As a rule, the CSS will occasionally pay the CSP for the services they have received. The next step is to create a SLA that spells out the conditions that both the CSP and the CSS must meet at all times or face the possibility of disciplinary action. Only the SLA, which details the whole service plan, service scope, scalability, outages, and incident response mechanism, is legally enforceable between the CSP and the CSS. A service variation (such as under-provisioned services) in terms of the agreed upon number of processors, memory, storage, bandwidth allocation, and service uptime assurances might be harmful to the CSS if the CSP provides unfair or degraded services to consumers utilizing the CSS's application.[97] Let's pretend for a moment that the CSP has a sensor network up and running to keep tabs on the various SLA assurances. For their effort if the SLA is breached, the CSS may want compensation.

As a quick recap, here's what's happened: The time frame is divided into "windows," or periods of the same duration (i.e., to capture the notion of the period). Afterward, the CSP communicates round c's worth of sensor data to the CSS towards the round's ending. The sensor data collects all the information on the SLA's numerous properties. The CSS makes an advance payment for cycle c + 1 if the SLA is met. However, the CSS has the right to seek compensation from the CSP in the event of a SLA violation in round c (for example, by appropriately decreasing its pre-payment for cycle c + 1 services). A CSP is expected to pay a CSS for its losses if it is provided with sufficient evidence of the CSP's SLA breaches.

Therefore, the CSS has the onus of evidence, in the form of demonstrating a SLA breach. Since the CSS could not provide all the necessary details, proving such claims might be extremely difficult. Even if proof is located, the procedure is laborious and time-consuming, which hurts the CSS. Therefore, the CSS would benefit from automating this SLA enforcement procedure. In particular, automating the process of collecting and presenting evidence would be useful for everybody involved. Since a loss-averse CSS will only pay a CSP in advance for cycle c + 1 services if the CSP has met the agreed-upon SLA for cycle c, SLA enforcement is presented as a fair transaction in which both parties benefit. The reasonable exchange also includes a mechanism for resolving disputes, so long as both parties agree that SLA breaches happened during period c, they will each receive a token of resolution that confirms the occurrence of those violations.

Thus, we define SLA Enforcement as the problem we address in the study as follows:

Definition 1: Putting Service Level Agreements into Effect. Incorporating the following into a single SLA S

- A collection of CSP sensors, abbreviated ICSP,
- An ICSS-denoted line item for "paying" the CSP,

- A standard called "ΣCSP" that defines the valid sensor readings for each ICSP sensor and their associated error bars.
- A ΣCSS guideline outlining the acceptable range of values for ICSS,
- A check mechanism called VICSP, which ensures that each sensor's reading is in accordance with the ΣCSP standard.
- VICSS, a mechanism for checking whether ICSS is consistent with ΣCSS;
- A time window TE that represents the temporal range during which sensor readings may be relied upon,

If protocol A satisfies the following requirements, then A may be used to enforce service level agreements.

- Fairness: When A terminates, then either (i) CSP receives ICSS and V_{CSS}(ICSS) and CSS receives ICSP and V_{CSP} (I_{CSP}) or (ii) if $\neg V_{CSP}$ (ICSP) then CSP does not receive ICSS but both CSP and CSS receive a resolution token that also contains the SLA contract S between CSS and CSP.
- Timeliness: It is guaranteed that A will end before TE.
- Non-repudiation: When A ends in (i), both CSP and CSS are able to independently prove that CSP is the origin of I_{CSS} and ICSP is the origin of CSS. If A leads to (ii), then both CSP and CSS have independent proofs of where their resolution token came from and what it contains.

4.3.2. Proposed Architecture

The picture describes a conceptual model that was developed after researchers learned about the SLA enforcement challenge and its accompanying gaps in an abstract. This theoretical framework is predicated firstly on a customer-service specialist (CSS) and a service provider (CSP). These two parties make up a two-party service exchange paradigm, with the service orientation set up to communicate synchronously with a third, crucial party—the trusted third party (TTP) who oversees and mediates the transaction. In synchronous communication, each event has a time limit, so if a response isn't received within that time, an alert will be triggered. Our algorithm also details how this design manages SLA enforcement by reducing the risks, vulnerabilities, and unfair exchange events that would otherwise arise. Detailed descriptions and demonstrations of the functions and interactions of several serving components, including SLA(E) and FE, are provided.

The next critical step was to determine how this theoretical framework might be put to practical use in a controlled experiment. We are integrating it with Microsoft Azure and other cloud platforms to monitor and track the actions of all involved parties. When we deploy CSS, CSP, and the TTP as virtual entities, we can study and verify how their constituent parts interact based on how those parts are configured in response to system specifications, operating environments, and other limitations. In the following, we discuss and quantify the ways in which our fair exchange-based in-line TTP implementation manages the service exchange to provide dependable and resilient automatic SLA enforcement.[81]

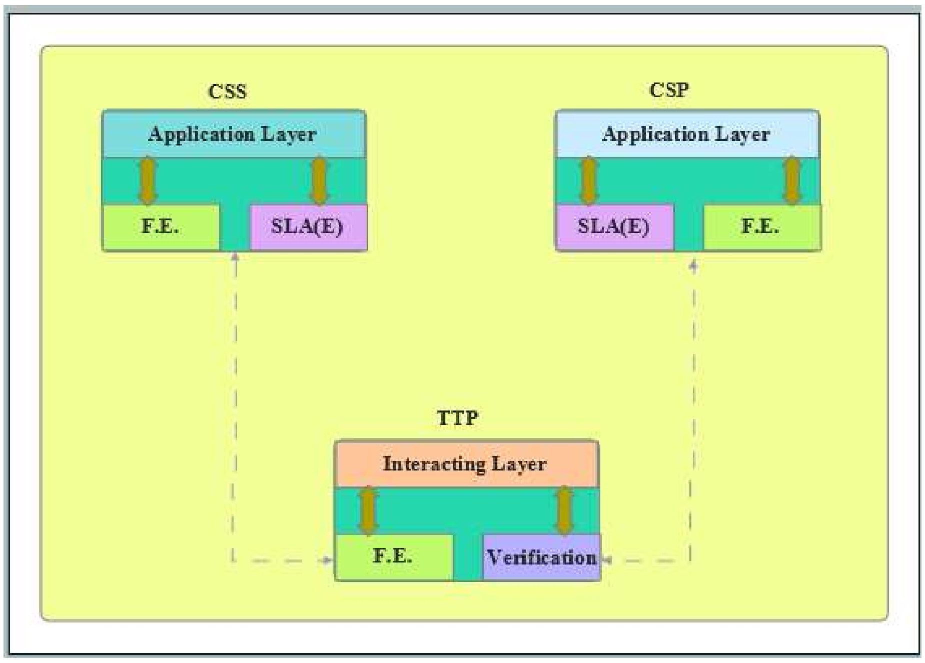


Figure 4.3: Proposed Architecture (I) (when participants are loss averse)

In Figure 4.3, a fair exchange component (i.e., a solution to the fair exchange issue) aids a SLA enforcement component. The idea behind this is that the SLA enforcement module will provide values or parameters to the fair exchange module. When the fair exchange components have the right information, they'll notify the TTP, CSP, or CSS to begin implementing the respective sub protocol (normal, resolve, or abort). The fair exchange subsystem of either the CSP or the CSS must confirm the swap has taken place. While the SLA enforcement module of CSS verifies that sensor values (reported by the fair exchange component) are within the SLA, the fair exchange component ensures that the agreed set of sensors (i.e., ICSP) is delivered.

However, it's difficult to enforce SLAs since the CSS often has to "pay" before the CSP begins providing services to the CSS. In the event of a disagreement, the CSS is responsible for persuading the CSP that a SLA violation has occurred. In such instances, the CSP and the CSS may still have disagreements for many reasons, such as the CSP disputing the evidence's origins (i.e., sensors). The extra work required of the CSS is the issue in this scenario. Additionally, in a typical fair exchange solution, both parties initiate the protocol simultaneously by exchanging messages directly with each other; the TTP is not involved in either the successful exchange or the usual termination of these communications. However, the SLA issue is asymmetric since the CSS must pay in advance for future services while the CSP may return sensor results only towards the end of the time window for continuous service supply.[82]

4.3.2.1. Architectural Components

Our suggested architecture includes three actors—the CSP, its users, and the sensors it employs—all of whom are external to the SLP enforcement problem but who are nonetheless updated by our proposal. (i) CSP, (ii) CSS, and (iii) TTP are the three players.

Table 4.2: Fair exchange protocol with post-exchange dispute resolution

Step	Protocol	Messages from /to PA	Message from/to PB
1	Setup	Φ(A)= eKA(IA)	Φ (B)=eKB(IB)
2	E	MA = eKB(L,A,H(N), ΦA);PA→ PB	MB = eKA(L,B,H(N),Φ B);PB→ PA

3	E	AckA(B) =(L,A,H(N),H(MB) my ack) : PA → PB	AckB(A) =(L,B,H(N),H(MA) my ack) : PB → PA
4	R	ResA =(L,A,H(N),MA,MB Resolve Req): PA → TTP	ResB=(L,B,H(N),MB,MA Resolve Req) : PB → TTP
5	A	ReqA =(L,A,H(N),MA Abort Req) : PA → TTP	ReqB=(L,B,H(N),MB Abort Req) : PB → TTP
6	R	MTTP(A) =(L,TTP,H(N),MB myack): TTP → PA	MTTP(B) =(L,TTP,H(N),MA my ack): TTP → PB
7	A	Abort TTP(A) =(L,TTP,H(N),Abort granted A): TTP → PA	Abort TTP(B) =(L,TTP,H(N),Abort granted B): TTP → PB

- **CSS and CSP**

Both of these parties need to have the SLA strictly enforced. Therefore, these two locations have two crucial features: Both a Service Level Agreement Enforcement (SLA-E) and Fair Exchange (FE) feature are included. With regards to the fair exchange problem, it is the FE component that is in charge of execution, and it is the SLA-E component that either sends or receives data in regards to SLAs. Our SLA enforcement procedure will be shown in two forms, all of which are modeled after the methodology in Table 4.2. I data exchanged between the SLA-E and FE, and (ii) communications between individual FEs.

Table 4.3: SLA Initialization

Step	Message from SLA-E to FE	Message from FE to SLA-E
1	ΨCSS =SigCSS(S,VCSS)	L
2	ΨCSP = SigCSP(S,VCSP)	L

Table 4.4: SLA Enforcement setup phase - Fair Exchange Component

Step	Protocol	Messages from /to CSS	Message from/to CSP
2	Setup	MSLA = (CSS,CSP, ΨCSS) : CSS →TTP	MSLA = (CSP,CSS, ΨCSP) : CSP →TTP
3	Setup	SigTTP= (CSS,CSP,H(N),L,TE) : TTP →CSS	SigTTP= (CSP,CSS,H(N),L,TE) : TTP →CSP

- **TTP**

Since the TTP's only function is to enforce the fair exchange issue, it does not include a SLA-E component. Therefore, the TTP's message exchanges are the responsibility of the FE part.

4.3.3. SLA Enforcement Protocol

We outline the procedure for SLA enforcement below.

- **Initialization and Setup**

The startup procedure is now being detailed.

SLA-E, FE at CSS and CSP: The SLA contract S agreed upon by the CSP and CSS must initially be sent to the TTP, who may then retain it in storage until the outcome of any potential disputes. In order to accomplish this, the CSS (or CSP) SLA enforcement section feeds the following information to the fair exchange section: To wit: I S and (ii) VCSS (resp. VCSP). Table 4.3 details the values that are signed off by the SLA enforcement section and forwarded to the fair exchange section. Upon receiving these parameters, the fair exchange component will proceed with the initialization procedure outlined in Table 4.4. The message carries a nonce that identifies the current "round" of an exchange by itself. The SLA enforcement module is informed of the value L (step 1), which is a label that ties to the SLA S at the TTP, from the fair exchange component at the conclusion of the execution.

FE at TTP: After receiving signals from CSS and CSP, the FE component TTP verifies that the SLAs are consistent. If they do, it gives the CSP and CSS an identifying label for the SLA, denoted by the letter L. The TTP must use the label L to refer to the SLA in all subsequent communications.

Table 4.5: Message Exchange between SLA-E and FE during Normal Exchange

Step	Message from SLA-E to FE	Message from FE to SLA-E
4	at CSS -(L,TTP, ΦCSS =SigCSS(ICSS)) at CSP - (L,CSS,Type equation here.Φ CSP =SigCSP(ICSP))	(L , ΦCSP) (L ,ΦCSS)
5	at CSS - (L,TTP,CSP ,my ack) at CSP - (L,CSS,CSS,my ack)	(L , CSP ,TE , my ack)

Table 4.6: Fair exchange protocol during normal exchange

Step	Messages from /to CSS	Message from/to CSP
6	MCSS = e K T T P(L,CSS,H(N), ΦCSS) : CSS →TTP	MCSP = e K T T P(L,CSP,H(N), ΦCSP) : CSP → CSS
7	AckCSS(TTP)= (L,CSS,H(N),H(MCSP) ,my ack) : CSS TTP	AckTTP(CSP)= (L,CSP,H(N), ΦCSS , my ack) : CSS TTP
8		AckCSP (CSS) = (L,CSP, TE ,H(N), H(ΦCSS) my ack) : CSP → CSS

Both the CSS and the CSP's FE components, upon receiving the message from the TTP, forward the label L to the receiving party, as this is the only piece of information related to the SLA, while holding on to TE, which is the time interval within which the exchange must be completed.

- **Successful Items Exchange**

Table 4.7: Message Exchange between SLA-E and FE during Dispute Resolution

Step	Message from SLA-E to FE	Message from FE to SLA-E
9	at CSS - (L,TTP,CSP, Ψ CSS)	(L,CSP,H(N),S)
10	at CSP - (L,CSS,H(N),S)	at CSP - (L,TTP,H(N), Ψ CSP)

After everything has been set up, there follows a round of item trading. We are not concerned with the precise value of TE in this study; nonetheless, it is possible that TE is a month, given that the CSS renews its CSP membership every month. Once startup is complete, CSS will be unable to use CSP facilities without first making a "deposit," or submitting his item through the ICSS system. However, as was previously established, the approach is unfair to the CSS. Instead of sending ICSS to CSP, he contacts the TTP. Then, before to TE, CSS will send a request to the CSP to retrieve the sensor values in order to determine if the SLA has been met. When the SLA is met, CSS notifies TTP that the SLA has been agreed upon, and TTP releases the "payment" to the CSP.

Table 4.8: Fair Exchange Protocol during Dispute Resolution

Step	Message from/to CSS	Message from/to CSP
11	ResCSS(TTP)=(L,CSS,H(N),MCSS,MCSP,Resolve Req) : CSS → TTP	ResTTP (CSP) = (L,TTP,H(N), ΨCSP, Resolve Req) : TTP →CSS
12		ResCSP (CSS) = (L,CSP,H(N), ΨCSP, Resolve Req) : CSP →CSS

SLA-E, FE at CSS and CSP: In this case, the CSS's SLA-E subcomponent transfers a legally binding "payment" to its fair exchange subcomponent in the form of a signed letter of agreement (see Table 4.5, step 4). After receiving accurate sensor data from CSP, the FE of CSS will trade this item for it. The SLA-E at CSS provides the FE component with I a signed item, (ii) the label L that may be used to identify the SLA, and (iii) the address of the recipient (in this example, the TTP). When complete, the FE will relay this information to the TTP.

Ahead of the TE period's end, the SLA-E at CSP will send the signed sensor values (as per the SLA) to its FE component, along with the recipient's id (in this case, CSS). The FE section of the protocol makes the appropriate changes to the state variables, such as the round identifying label L and the nonce N for the sought-after SLA. This is then sent to the FE subsystem of the CSS, where a preliminary sensor validation is conducted (i.e., if the correct set has been delivered). If so, the CSS's SLA-E component receives the signed sensor values CSP and label L from the FE component and determines whether or not a SLA violation has occurred. Importantly, in the event of a SLA breach, any potential dispute settlement might take up to Δ time units to settle. Therefore, prior to TE - Δ, the CSP must send the sensor values it has collected to the CSS.[83]

The SLA-E subsystem at the CSS will send an acknowledgement to the FE subsystem once the SLA has been met. After this is complete, the FE component sends a notification to the TTP and includes a hashed replica of the sensor readings. Once this is accomplished, the TTP will send an acknowledgement message to the CSP's FE component, including the CSS's signed "payment". The CSP FE component then acks its CSS counterpart, and the payment is passed to the SLA-E component. The SLA-E component then acks the CSP FE component.

FE at TTP: By (i) getting the signed "payment" from the CSS (Table 4.6, step 6), (ii) receiving a confirmation from the CSS to proceed to pay the CSP for successful execution, and (iii) making the payment to the CSP on behalf of CSS, the FE component of the TTP plays a role in the successful exchange.

4.3.4. SLA Violation and Dispute Resolution

When a SLA violation has occurred the CSS will send a separate message than when there has been no violation.

SLA-E, FE at CSS and CSP: When an issue occurs, the CSS SLA-E team will transmit a copy of the original SLA contract to the SLA FE team as notification. The TTP then communicates this to the CSP, as indicated in step 11 of Table 4.8, and the FE of the CSP communicates the Resolve Req of the CSS to the SLA-E of the CSP, as shown in step 10 of Table 4.7. At last, the CSP's SLA-E component reports the violation to the CSP's FE component, and the FE component reports it to the CSS.

FE at TTP:TTP is involved in the SLA violation and dispute resolution process in two ways: (i) when it receives a message from the CSS informing it of a SLA violation, (ii) when it verifies the validity of the CSS's complaint and, if valid, notifies the CSP by sending a copy of the contract, rather than the "payment."

4.3.4.1. Correctness

We provide evidence supporting the validity of the currently-accepted procedure for SLA enforcement. All three of these conditions must be met for the SLA enforcement issue to be considered solved. (i) fairness, (ii) termination, and (iii) non-repudiation.

4.3.4.1.1. Fairness

This evidence has two parts: (i) when the protocol ends without a disagreement resolution, and (ii) when a dispute resolution occurs.

No dispute Resolution:

Because each participant's SLA- E is already in place when the exchange procedure begins P □ {CSS, CSP} sends an IP for a signed item to its corresponding FE part; this ensures fairness at the SLA-E level because fair exchange satisfies fairness.

Dispute Resolution:

For a disagreement to enter into dispute resolution at the CSS, VCSP (ICSP) status must first be achieved. After then, according the protocol, the CSS's SLA-E must inform its FE subcomponent of the violation by sending along a signed copy of the contract (Table 4.7, step 9). When TTP receives this message from the CSS, it does not relay ICSS to CSP, but it does inform ψCSP, detailing the outcome of the disagreement. We assume a loss-averse user so the CSP will know if there has been a SLA breach.

4.3.4.1.2. Termination

All four steps of a typical or successful exchange occur when CSS gets ICSP from CSP including sensor readings, sends ack to TTP, TTP forwards ICSS and ack to CSP, and CSS receives ack from CSP. Assuming the restriction on (synchronous) communication delays is D, the usual exchange may finish in 4D time after CSP has commenced delivering ICSP (after accounting for time required for local computations). Be aware that the CSP can continue providing cycle c + 1 services to the CSS in tandem with the final communication step.

Within 4D time, the CSP commences the transmission of the ICSP, the CSS sends back a Resolve Req to the TTP, the TTP resolves the request and alerts the CSP, and the CSP informs the CSS of a "adjusted payment" for cycle c + 1 services owing to SLA violations in cycle c. Thus, $\Delta = 4D$.

After a dispute is resolved, the CSS may resume execution of the fair exchange with updated payment conditions. Upon receipt of the amended payment from CSS (Step 6 of Table 4.6), TTP will send its ack to CSP (Step 7 of Table 4.6), and CSP will resume delivering services for cycle c + 1. (in Step 8 of Table 4.6). To guarantee that payment from CSS, notwithstanding any SLA breaches, reaches CSP by TE, CSP shall commence transmission of ICSP for cycle c no later than TE 6D, assuming that the preceding two phases would end within D time.

4.3.4.1.3. Non-repudiation

The concept of non-repudiation describes the impossibility of the CSP or CSS to back out of a promise or obligation. For instance, the CSP has no legal standing to assert that the sensor values are not its work. For this application, we must demonstrate that the CSP can verify that the sensor data provided by the ICSP come directly from the CSP. Fourth in Table 4.5, the SLA-E signs ICSS before sending it to the CSP's FE subcomponent. Non-repudiation is guaranteed since the signed item is sent by each participant's (CSS and CSP) FE to their corresponding SLA-E component.

In this paper, we demonstrate that the SLA-E protocol, when used in conjunction with the FE module, satisfies all of the SLA specifications.[84]

4.3.5. Protocol Implementation

The word "protocol" is used in the field of distributed computing to refer to a predefined set of rules and formats for exchanging information across different parts of a system. The exchange of messages and the data format, together with their mutually agreed upon specifications, are crucial building blocks of every protocol. Whereas, "a mechanism by which two computers coordinate their communications" OR "an established set of norms by which two computers or communication devices authenticate the structure and substance of the messages transmitted" best describes a communication protocol. This subsection provides a more in-depth analysis of our suggested approach. The architecture, range, constraints, attributes, method, and flow control paradigms of our FEP fair exchange protocol-based solution will be revealed. In addition, we will discuss the testing environment, the potential connections between the dots, the design's components, and the difficulties encountered in a variety of computer settings.

To ensure that our protocol works as intended, we have installed several modules that mimic the behavior of software, networks, and traffic in order to test its ability to enforce SLAs automatically in the cloud. This implementation also shows how the several processes and procedures are bound in a systematic way to manage the three main stages of a transaction, including the message exchange, message abort, and dispute resolution.

This study's primary goal was to take our previously suggested conceptual model and turn it into a working prototype, complete with a detailed yet easy-to-understand explanation of how each component works. Our notion was conceptualised as a high-level illustration of the protocol's operational structure and the predicted behaviour of its associated system. In practice, however, implementation often includes the physical, logical, and service regimes of the protocol in order to accommodate several different users, such as the CSP, the CSS, and the TTP.

4.3.5.1. Physical Regimes

Virtualization has significantly altered the IT services industry. Proponents' claims that cloud computing is superior than on-premises infrastructure in terms of provisioning, deployment, maintenance, on-demand serviceability, affordability, and other criteria are hard to refute. From the perspective of cloud computing, the "physical computing entities" are the virtualized computing resources given to a service subscriber, for example IaaS. Cloud users that pay for a premium IaaS service have direct access to the CSS data centres, where they may use a variety of resources, including storage, networking, computation, load balancing, and security-related technologies.

4.3.5.2. Regimes

When it comes to business needs, a logical regime in the client/server architecture encompasses a wide range of operations, specialized applications, and unique pieces of software. These responsibilities may involve modifying data in many ways, such as transmitting and receiving data, processing, and calculating it, storing it, or even rerouting it from one or more clients to the appropriate serving nodes in a distributed system environment (such as the cloud).

Such rationally conceived platforms can integrate the anticipated interactions between various data/security controllers and business processes, allowing them to carry out tasks in parallel. Some cases include gaining entry to data, transferring data, processing data in batches, creating intercommunication, integrating networks, and even ending services entirely. These technologies also make it easy to save certain details for later use as a point of reference when referring to a different gadget or service module.

4.3.5.3. Services Regimes

The service scope defines the parameters within which the service must be provided by all partners for the company to generate a sufficient return on investment. Limitations and possibilities in terms of deployment scope, operational controls, service dimensions, and optics When a business decides to outsource its services to a CSP or a cloud service consumer, the CSP or the consumer may choose taxonomy, service monitoring, and service reporting (CSS). Definition, ownership, fault-tolerance, rectification, and, in the worst event, termination of service pertaining to defects are also covered. Scope of services is as important as price in determining other factors such as possible conflicts of interest, length of contract, frequency of renewals, method of dispute resolution, and policy for escalation.[85]

4.3.5.4. Participants

An overview of the connections between our four business participants to provide and deliver cloud-based services is shown in Figure 2. This new architectural perspective improves the ability to learn about the operational scope of the participants' computing estates. It also sheds light on the provisioning of certain fundamental modules, like an end-user estate simulator built with Apache JMeter. CSP, CSS, and the TTP all benefit from the FEP controller's incorporation. These parties, excluding end-users who have no use for this component, have access to the Master SLA repository for the purpose of cross-referencing. Since TTP isn't necessary, the SLA-E module would only be incorporated into the CSP and CSS.

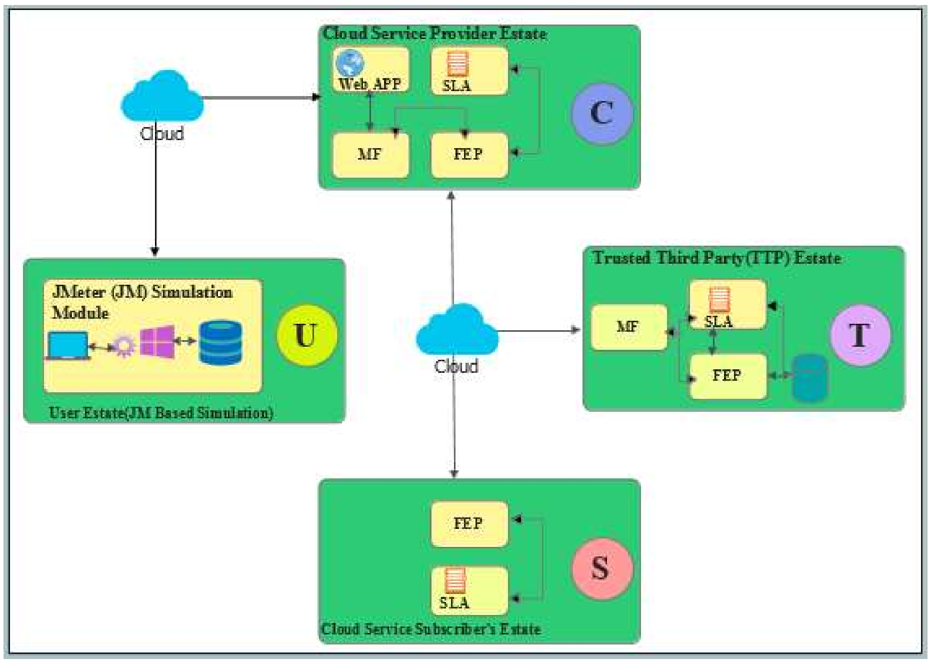

Cloud Service Provider Estate
Web APP
SLA
MF
FEP
C
Cloud
JMeter (JM) Simulation Module
U
User Estate(JM Based Simulation)
Cloud
Trusted Third Party(TTP) Estate
MF
SLA
FEP
T
FEP
SLA
S
Cloud Service Subscriber's Estate

Figure 4.4: Architecture's Operational Interaction - Preview

4.3.6. Protocol Mechanisms

This section elucidates the fundamental capabilities of our protocol and how they work together. All necessary operational conditions (such as SLOs, dispute resolution, service termination, etc.) must be defined and met before the protocol initializes core components like message exchange. How are different parts of the service, such the protocol's execution order, to be tracked? What factors into the decision to trade messages when transaction parties such as CSP, CSS, and TTP are exchanging digital services against agreed upon financial obligations? Another noteworthy phenomena is the establishment of defined anomaly detection techniques. Latency, poor performance, and a lack of auditing are just a few of the service overheads that may be avoided with proper attention to the protocol's flow and error management, which otherwise could lead to inconsistent and unpredictable SLA breaches.

- **Protocol's Sequence Control**

The order of steps in a protocol is crucial. This correlates with the established logical sequence of how communications (to/from) are delivered and received. To avoid any potential for misinterpretation while sending messages or processing requests from clients, this system uses an algorithm to generate a one-of-a-kind reference number, or unique message identification number (UMIN), for each message. In addition, these protocols provide a one-of-a-kind message exchange signature for precise, exact, and exhaustive digital forensics tracking. UMIN methods for recognizing authentication, reconciliation transaction completeness, and audit ability (missing/disputed transactions) prior to message tagging are widely agreed upon by service providers and their clients to guarantee correct accounting.

Priority classification of messages, as decided upon by some or all of the participants, may also be established with the use of this exercise. In our test implementation, where user assets are modeled as (U), Apache JMeter generates a unique identifier (e.g., a thread reference number or an e-Tag) for every message sent and received. Thus, a UMIN may be defined. It's an MF that logs incoming and outgoing messages with timestamps and a RI/RO. Eventually; it transmits each message to the server hosting the relevant online service. It then gets the reply from the CSP's web service, and follows the same procedure as previously, recording the RI and RO, before sending the reply to the message's originator, the U. If a dispute resolution situation occurs, the whole sequence of data can be saved so that its provenance can be investigated from the time it was communicated across various entities.

4.3.6.1. Protocol‘s Flow Control

This function controls the status of being choked off when an excessive influx of data from one computer to another (the Client) causes severe bottlenecks in service. To prevent this situation from occurring when request handling and processing capabilities become stuck, the responding node must implement flow controls. To prevent a denial-of-service attack, traffic must be carefully monitored and measured, process, query, and compute handlers must be outsourced, and dedicated resources must be established ahead of time. Flow control sensors are primarily responsible for gathering 5-tuple data, including source IP, source port, destination IP, and destination port, as well as layer 4 protocol information. Depending on the availability of resources, the needs of the company, and the goals of the service deployment, blocking and non-blocking criteria for message transmission may be used to govern the flow. Some of the states that help to clarify message flow management are as follows: - [86]

Time Oriented Flow Control:

The CSP and CSS have agreed on a certain window of time for this regulation to take effect. SLAs, for instance, do reveal what kinds of computer work may be done during peak and off-peak hours. These metered services are meant to facilitate easy traffic flows to and from CSP data centers.[130] There should be as little disparity as possible between subscribers' access to services and resources by enforcing these time limits on customers. If a CSP's flow control monitoring sensors detect an anomaly, the CSP may reset, release, or terminate the service connection at any moment. What sort of financial promises are being made by the CSP and the CSS also has an impact on the flow control. The time orientation is also used to monitor SLA provisioning for cloud services like AWS Amazon.

Performance Oriented Flow Control:

The efficiency with which a cloud service performs is essential. For example, while evaluating a service's effectiveness, several factors, including bandwidth, response time, latency, and availability, are considered.

Security Oriented Flow Control:

This part of the protocol flow control is in charge of protecting the underlying system. From a service security perspective, traffic audits are extremely thorough. Communication is dependent on the flow and behavior of traffic. Data centre traffic is monitored and analyzed on a regular basis using a signatures repository. For instance, Amazon Web Services (AWS) and other cloud service providers include a monitoring feature that allows for the capture and monitoring of traffic flow logs within a virtual private cloud (VPC). Any abnormal traffic patterns, such as those caused by outages, security breaches, or even a downgrade in service, can be detected and investigated with the help of this feature.

4.3.6.2. Protocol’s Error Control

Error detection in digital communication refers to the process through which a protocol identifies and depend on other features that ensure the message's integrity intrinsically. It determines whether a message exchange between two parties is secure on unreliable networks like the internet by checking the authenticity, validation, and verification elements of the sender and the receiver. Since error handling has the potential to turn a service's success into failure, it is expected to receive higher priority. Task failure, communication breakdown, processing timeouts, and unauthorized changes to the system can all cause disruptions in service and constitute a breach of the SLA. Messages can be checked in an instant for errors, integrity, accidentally injected faults, and other irregularities using a number of different schemes, including checksums, parity checks, and cyclic redundancy checks (CRC) -enabled appliances and devices. In the world of distributed systems, a communication protocol needs to have several safeguards in place in case of an error. The first attempt to send a message fails before it even reaches the end user's system. The second transmission leaves the domain of the end user but is never received by the domain of the CSP. In the end, there's a situation where the CSP handles the requests, but those requests for whatever reason never leave the CSP's network and instead reach the end user.

Let's take a quick look at how AWS deals with errors. They have classified the mistakes primarily into two groups. The first set of errors is Client-side errors (error codes 4**), which occur when AWS APIs refuse to process a request from the client. The credentials, parameters, and agreed-upon configurations could be authenticated, verified, or validated. Error codes 5** are associated with problems in AWS's service environment, such as its networks, compute, or storage, and indicate that the nodes responsible for fulfilling the client's request(s) either failed to do so or took an excessively long time to do so. When waiting for this excessive amount of time to process those requests, SLAs are broken. In the event of a disagreement over service delivery, the CSP should use the results of a forensics investigation to determine whether or not to issue service credits to the affected customers.The Amazon Client Exceptions and Amazon Service Exceptions in a cloud service like AWS may reveal the cause of a service failure.[87]

4.3.7. The Architecture Scope

The scope of architecture includes the most essential service and operational responsibilities. The purpose of this article is to shed light on a few operational expectations and the conditions under which our procedure might exhibit unexpected behavior. Although it is not necessary to detail how each individual component operates, we must describe some aspects of their scope. Perimeter-configurations at both the C and application-level settings at the U must be agreed upon by all parties involved in the protocol implementation. The proof of concept will show the protocol's operational and authoritative capabilities, such as monitoring, aborting, stopping, or even temporarily terminating transactions (in the case of unfair and uninvited service delivery).When a disagreement between a CSP and a CSS is designated as unresolved, the primary goal of resolving disputes becomes active. All parties are familiar with their assigned levels of trust, the services they provide, and their roles in the integration of the FEP modules.

Until the conclusion of the contract period, this architecture will treat as an auxiliary-module any portion of any participant's environment that affects SLA enforcement or limits service monitoring. Assumptions will also be taken into account, including but not limited to the following: all participants have successfully implemented all sub-protocols, processes, procedures, system configurations, and segregated service monitoring controls; and the master configuration has been successfully implemented. TTP's FEP master node accurately stores SLA data. By verifying, validating, and persuasively collaborating with SLA-E modules at C and S, the FEP module will be able to impose an automatic, perfect, and trustworthy dispute settlement. The ability to record service-level agreement measurements (such response time, bandwidth, latency, etc.) and store them locally is also important to the protocol's design.

Referring to Figure 4.4, it should be clear that despite the fact that all parties involved (C.U.T.S.) are autonomous computing entities, they are nonetheless connected thanks to the FEP component, which synchronously analyses whether or not message exchanges comply with SLAs. CSPs usually do establish some joint accountability for maintaining, protecting, and operating the services they provide. Additional adversaries in SaaS, such as those related to the data in transit (insider/external cyber-attacks, malicious participants, intentional/accidental service compromises, service degrading, etc.), between the end user's estate and the trust boundaries of the service provider, are beyond the scope of this paper. There are many potential threats in global topologies like the internet, which use many third parties as intermediary hosting services (middle-tier service carriers) to transport data but will be ignored for the time being.

4.3.7.1. JM‘s Scope

Since this article presents a prototype, we decided to utilize Apache's JMeter to emulate the user's environment. The test plan is set up to simulate minimal HTTP traffic going to a predetermined IP and port on a distant MF. Adjusting the settings of a JMeter test plan can demonstrate the diversity of users' actions. The primary goal of this implementation was to solicit comments from site visitors seeking to make use of a cloud service. JMeter's job is to move these messages through the MF protocol to their eventual destination (say, a cloud service) and back again. As part of its application-level duties, JMeter may also be required to keep track of other crucial data. JMeter would log HTTP status codes based on the status of the service provisioning agreements, availability, and uptime. It continues sending messages until either the client's allotted time has elapsed or a local termination signal is issued for the service.

4.3.7.2. MF‘s Scope

The primary function of the MF is to receive messages, append a timestamp to them, and then deliver them to the intended cloud service. The queries will be processed by the cloud service, and the MF will get the results. Timestamped responses are then sent back to the original senders by the MF. For the next term's upfront payment to the cloud service provider, MF first determines the response time and records the whole transmission. The CSP then regularly relays this information to the CSS so that SLA compliance transactions may be recorded.

We also need a node that relays messages to their intended recipients, called a message forwarder (MF) (s). As we've seen, the FEP module serves as the brains behind the operation, calculating the quality of service with the idea of service level agreement enforcement at its core. This module is placed on the ends of the CSP, CSS, and TTP in such a way that their modules are interconnected and can initiate communication among the participants in the event of a violation or dispute enquiry.

4.3.7.3. FEP‘s Scope

The FEP module is the nerve center for SLA enforcement and quality of service evaluations. While CSP's response transmissions are taking place, it compares each message to the central configuration repository for SLA. The time, date, and status of sent and received messages along with their respective transaction identifiers and timestamps are the primary components of those statistics. CSP and CSS rely on MF to manage an assessment method that generates the ART. The FEP alerts the TTP using these ART. The FEP also covers communication channel operations including Exchange, Abort, Resume, and Terminate.[88]

4.3.7.4. SLA(E)‘s Scope

SLA(E) not only collaborates with the FEP to reconcile the SLA, but also brings the SLA between the CSP and the CSS up-to-date with each new service negotiation. Master configuration file with SLA metrics has been reviewed and accepted by TTP (such as the agreed response time).

4.3.8. The Protocol Limitation

Certain restrictions and operational constraints must be taken into account throughout the design and implementation of a protocol. In a decentralized SOA design, these vulnerabilities might have a serious effect on service delivery. To diagnose the issue, one must be familiar with every aspect of the service, from the perspective of both the client and the service provider. An operating system process may encounter trouble if the allocation of computing resources is altered, the configuration of the system is changed, access to the system is restricted unnecessarily, or more security burdens are placed on the system. Here is a rundown of what may be the most detrimental of these restraints, should they be relaxed:

- Assets Accessibility
- Metadata (e.g. web service metrics) file format
- Data Interporatability
- Timely Negotiations (SLA & SLOs)
- Conflicting Interfaceability
- Unpredictable Faults at Operational Boundaries
- Legal & Regulatory Constraints
- Monetary Limitations
- Clock Synchronization
- Unidentified Attackers & service compromises
- Involvement of another 3rd party/ service contractor

Most cloud service providers are fairly strict in their requirements for monitoring and collecting metadata pertaining to provisioned services. Since they employ a number of throttling techniques to limit resource sharing, it appears that they do provide some level of security monitoring for their web services, albeit on a very limited scale. There are options for working together to monitor, such as Amazon's CloudWatch and other third-party tools, but there are also limitations. How closely are things being watched? I would want to know what information is being gathered and how much. It is too unsafe for service subscribers to associate with 3rd parties, although certain SaaS systems enable integration with tools for precisely monitoring Application Performance Management (APM) such as AppNeta, Dynatrace, and DataDog through their APIs. Even though these organizations are able to monitor and gather data for their clients, they still can't promise that the subscriber's business needs for a certain service element will be satisfied. As a result, a service subscriber may wind up deploying too many tools to fulfill business needs, on top of the CSP's extra costs for monitoring and collecting these data. As some of the service pieces are created and controlled by the 3rd parties when messages are delivered or received, our protocol does encounter some of these constraints at specific computation stages. To guarantee a successful connection is created with a minimum failure probability, numerous service performance, scalability, and reliability factors are assessed and applied.

4.3.8.1.Architecture‘s On-premise Virtualized Deployment-SN1

Testing an entire architecture on a single computer is infeasible because of the strain placed on the system by the simultaneous processing of various, resource-intensive tasks, such as the constant influx of messages. A single CPU could only perform a limited set of instructions at a time, but a distributed environment, such as virtualization, may provide the necessary computational power and service resilience for a project's successful completion. Our strategy for testing the FEP-based architecture in different environments is shown in the following figure.

Cloud computing is highly dependent on virtualization, a core feature of many virtualization systems. By using these wonder frameworks, currently limited computing resources may be converted into multi-shared computing infrastructures for a wide range of commercial purposes and service providers. Between the software and the underlying computer hardware is an abstraction layer, which increases adaptability, reliability, and security while decreasing redundancy and cost. To make the most of the available features, we purchased many virtual machines.

By creating endpoint communication channels and assigning suitable privileges and settings, we permitted communication amongst our crucial transaction participants who each had their own isolated virtual machine. This configuration helps JM because it can send spoofed messages to a second virtual machine acting as a stand-in for an MF before redirecting the information to the target website. We've installed an application server on this web server (the CSP's node) to process requests from other clients and send their results back to the original client. All the nodes except from TTP have MF set up. Evaluating the necessary SLA metric ART was the primary function of the MF module on CSS. The selected CSS node would regularly provide SLA measurements as a (.csv file format) to the CSP for reconciliation and forensics, and each simulated HTTP request would be time stamped and recorded in a metadata repository. The reality of any SLA violations would be established by these periodic communications, allowing the afflicted party to be reimbursed via the fair exchange protocol. QoS also gets analyzed thus if the mentioned service is deemed SLA compliant then just the next service term will be agreed upon by paying the upfront costs, else, the service will be cancelled.[89]

When this experience, which uses a small number of dedicated on-premise virtual machines, is being executed, a lag in the expected ART is seen since the system's resources are being used above their capacity, and this leads to ART anomaly alerts being sent. As a side effect of this test, we discovered that the response time is erratic and inconsistent, most likely as a result of the system being overworked. As more and more of a system's resources are used simultaneously, less and fewer are set aside for pending JVM threads. Even though we followed Apache's rules and other best practises when configuring the number of mock users on our JM test plan, the local machine's resource management seems to be a bottleneck for the VMs running the example web application's efficient response to messages.

4.3.8.2. Architecture‘s Cloud Deployment-SN2

Business as a whole has been profoundly impacted by cloud computing ever since it first became available to the public. Migrating data assets including infrastructure, development platforms, and web-based applications may be very beneficial for businesses of all sizes. However, there are also considerable challenges associated with this course of action.

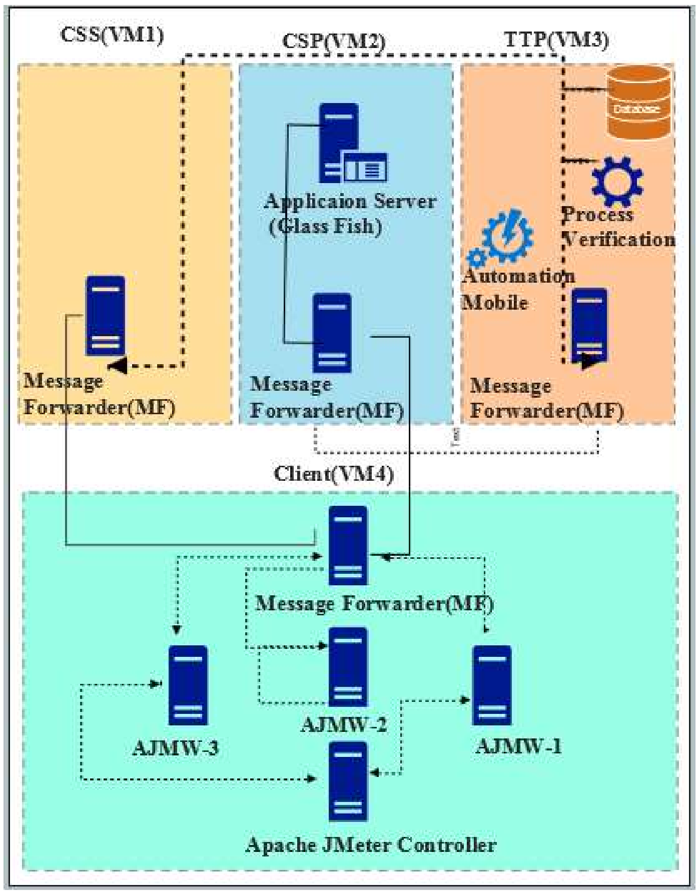


Figure 4.5: Architecture's Cloud Deployment on Azure

Figure 5.6 below illustrates the architecture with respect to the specified procedures running on each node. The protocol may perform any number of technical procedures, depending on the current stage and health of the participants. The terms "Exchange," "Abort," and "Resolve" are all considered as possible intermediate steps. In order to ensure that processes run on time and that nodes communicate as intended, most operational conditions are specified during the setup/initialization phase of the service. Under this paradigm, all four participants in the protocol communicate with one another, with TTP serving as the service monitor and observer between CSS and CSP.TTP's SLA repository may also maintain SLAs for other clients, thus any unexpected behavior will trigger artefacts verification against the master SLA, as mentioned in the preceding interactions. Time-based synchronization limits the scope of both service and process execution. Remaining primarily at the TTP, the FEP server rigorously performs reconciliation against the master Service Level Agreement (SLA), which has been communicated to and acknowledged by both the CSP and the CSS, and which tracks the evolution of service exchange processes throughout the service lifecycle. Thus, the end-to-end fairness is ensured by these interdependent and coordinated procedures.

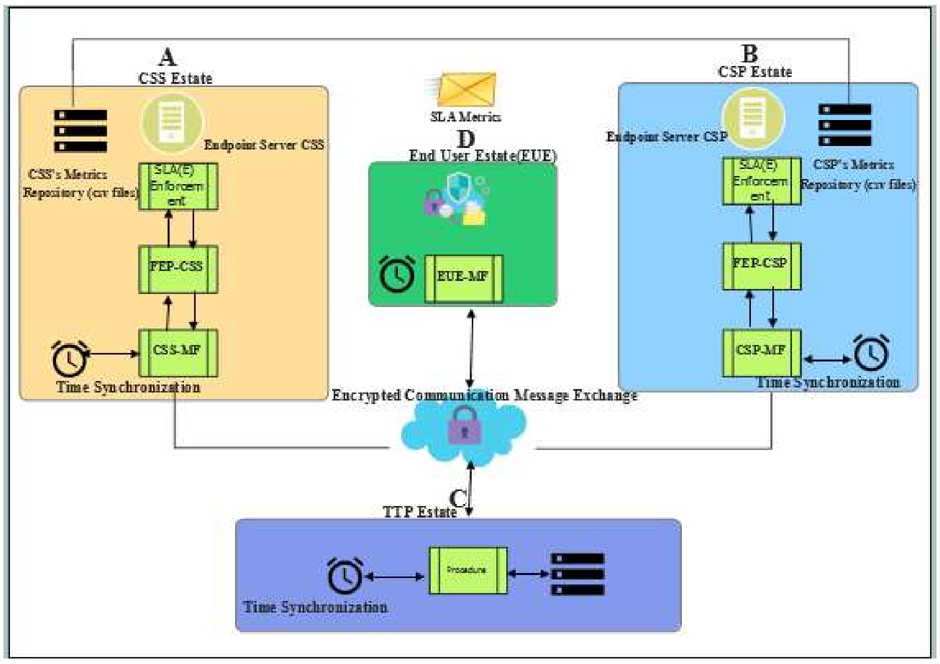


Figure 4.6: Process Execution

These problems may be broken down into three broad categories: technological, operational, and security. Interoperability across resources, granularity, service orientation, commercial concerns, security, performance, and service quality are just a few of the many areas where problems arise (QoS). We have spoken about cloud computing(CC) & SLA Implications and the important service pieces and characteristics that go along with it, as well as how they all come together in a mutually agreed upon service level agreement. Our final objective with this sandbox is to demonstrate that we can guarantee SLA compliance via a reasonable exchange mechanism.

Although there are several cloud service providers, including Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure itself, we chose Microsoft

Azure as our main test-bed. Elastic computing resources are Azure-hosted virtual machines. These tools include instant cloud provisioning, allowing clients total freedom of choice among several customer-focused cloud services and Cloud Application Programming Interfaces (CAPIs). On this context, "cloud infrastructure deployment" might refer to anything from a small-scale experiment in the cloud to a massive enterprise-level application or other strategic automation solution. Unlike traditional on-premises computer installations, most CSPs provide platforms with low setup, maximum scalability, rapid set-up time, security, and strong failure resilience.

We assume a realistic cloud service scenario in which CSS A leases infrastructure from CSP (B) and, in return for payment and other commitments, provides a web service to its client base (D). We've learned in the past that the Service Level Agreement (SLA) contains the full breakdown of the service delivery plan, from deployment to Service Level Objectives (SLOs), Quality of Service (QoS), expected services, and compensation workouts in the event that the CSP doesn't meet the SLOs by a certain date or within a certain scale of available resources.

Our top priority is always the reliability of our results, particularly when our protocol's administrative handlers—like an Exchange, Abort, or Resolve—are activated, recognized, and reported. This is why we extensively tested our protocol with several cloud deployment options. Supports the growth of our backend monitoring and response nodes like JMeter and MF. To fictionalize mutually trusted autonomous SLA enforcement, our architecture would need to have a system configuration, inter-communication schemes, security composition, and fault tolerance that more closely resembles real-time SLA-based cloud service delivery with as few performance issues and other potential technical bottlenecks as possible. [89]

4.3.8.3. CSS Environment

In order to provide a realistic environment for our participants, we will need to start up four virtual machines: one for the client environment, one for the TCS, one for the TTP, and one for the content security service provider (CSP). This has been tested before on Amazon Web Services and other cloud resources, with the most current round of testing performed on the Microsoft Azure Cloud. See the part where we describe the infrastructure instances and the associated logical and service orientations, capabilities, and functions for each instance for additional information on how our design is implemented. Due to Azure's extensive collection of OS images, we decided to run Ubuntu 18.04LTS on a virtual machine with 2 vCPUs, 8 GB of RAM, and 30 GB of standard SSD storage for the CSS node. We did not expect a huge demand for CPU, memory, or disc I/O operations per second (IOPS) from our experiment, thus we did not feel it necessary to provide these high-end resources at this time. An Azure VM instance may have its operation started, stopped, restarted, and even terminated at any moment. Inasmuch as the CSS instance represents a centralized hosting server on which a number of strategic programmes and monitoring sensors have been put up, it would be beneficial to initially address these components.

First, I'd like to introduce you to our website, where you may learn more about our MF application that runs on the Java platform. In addition to mediating interactions between the CSP and the Client estate, it monitors key performance indicators including response times. All (incoming and outgoing) messages are also timestamped for later reconciliation purposes. To ensure appropriate service exchange, this protocol's principal administrative control will be based on the SLA. When a client sends an HTTP/HTTPS request to a server, the messages are sent in a synchronous system in a strictly chronological sequence. Most often, one of two models is used to organise the flow of messages: a synchronous, blocking model or an asynchronous, non-blocking model. Customers from Client Entity D using the CSP-provided web service should anticipate an HTTP status code of 2xx Success, 3xx Redirection, 4xx Client, or 5xx Server. We want the client to perform an availability check on the CSP service and get an HTTP status code of 200 when the service is initially made available as a proof of concept. Requests from clients to the hosting server and responses from the server back to the originating client are both common components of messages carrying metadata about HTTP/HTTPS requests; the relative proportions of these two types of requests, however, vary depending on the number of clients. The measurements are recorded in a.csv format in an Azure Storage Blob for future experimentation, and the same MF agent keeps track of them to acquire and test greater flexibility and resilience while testing our design.

For the time being, the focus of this work is on synchronous communication, so we are not utilizing any of the robust message queuing systems designed and well approved for message queuing based upon asynchronous communication, such as RabbitMQ or ActiveMQ. In the future, however, we hope to transition to an asynchronous communication model. Because of Pika's versatility, it may be utilized with either asynchronous or synchronous models.

Microsoft Azure is the finest platform from which to construct this environment since it provides a comprehensive service catalogue from which to select a variety of elastic services. They make it easier to quickly deploy cloud-based systems that provide computing, storage, networking, and other mission-critical services for a business. In order to provide a realistic environment for our participants, we will need to start up four virtual machines: one for the client environment, one for the TCS, one for the TTP, and one for the content security service provider (CSP). Fig. 5.3 depicts the three-pronged deployment of our architecture, which consists of the infrastructure estate, the actual virtual machine(s), and the multiple players in our architecture. The service fabric then shows how everything is connected once the logical deployment strategy has shown how everything talks to one another.

Our choices for the (virtual) infrastructure of a cloud experiment have been limited down to using either an Azure compute instance or a hosting architecture that gives us full control over all sandbox nodes. Eventually, we want to use Azure's compute service catalogue to containerize these services using choices like Azure Kubernetes Service (AKS) and application service (App Ser- vice), but for now, we're simply using virtual machines (VMs) for each of our testers. From Azure's extensive library of OS images, we settled on an instance series, which is a typical Ubuntu 18.04LTS VM with 2vcpus, 8GiB RAM, and 30GiB standard SSD, for the CSS node. It was unnecessary to provide premium resources such central processing unit (CPU), memory, and disc input/output operations per second (IOPS) at this time since neither the initial demand nor the predicted consumption of our experiment was expected to be very high. As a virtual machine, clients may deploy as many Azure VM instances as they need and manage them in the usual ways: starting, stopping, restarting, and even terminating. For our pilot project, we opted for Azure's free tier of instances, which is rife with restrictions on both functionality and maintenance. More free hours per instance per month may be operated with no or limited assistance in the business editions, which are designed for heavy use and include more features and higher resource requirements including 24x7 support, backup, monitoring, and other corporate maintenance rights.

Extra features like virtual network creation, NIC settings, and a network security group (NSG) to define allowed and forbidden states for our network traffic are implemented according to the service provider's guidelines and best practices.

SSH (Secure Shell) and RDP (Remote Desktop Protocol) allow for secure remote logins after an RSA public-private key combination and the IP address of the target node have been established (Remote Desktop). You may get remote access to the instance by downloading the necessary access keys, which works with the aforementioned protocols. Each VM instance is given both a public and private IP address for purposes of identifying and communicating with the instance.

Ubuntu was the ideal option for Microsoft's Azure platform because of its widespread use, versatility, scalability, and smooth operation. In the realm of cloud-based development and innovation platforms like Azure, it comes as no surprise that Linux is the platform of choice. This prototype's central server is built using an Ubuntu server image since it is flexible and compatible with the protocol we're utilizing. Providing processing and monitoring for all of CSS's provided cloud services for all of CSS's client estates, this controller node is where it all happens. However, in order to test how well the CSP delivers on the quality of service promised in the SLA, we will be concentrating our attention on a single customer and their deployed cloud services.

In this particular scenario, we are also making use of our Java-based MF module. This would eliminate the risk for delays and other network issues while decreasing the implementation cost. By picking up the data from the user's location, MF may forward it to the cloud. It's possible, as a bonus, that it will relay the user's signals back to the home where it was set up.[90]

Application Deployment

Unbelievably, AEB offers highly elastic web applications and other cloud-based services that let us to broaden our application footprint. Due to the platform's adaptability, enterprise-level applications built in languages such as Java, Python, Go, etc. may be readily deployed to a wide range of application server environments, such as Apache, IIS, Glassfish, etc. The AEB is capable of automating a wide range of operations, including but not limited to capacity provisioning, load balancing, multi-featured health monitoring, and rapid alerts transmission to a wide variety of recipient platforms (email, SMS, etc.). In addition, applications may be set up with auto-scaling triggers, allowing them to adjust their resource use based on their budget.

4.3.8.4. CSS Environment

A replica of SLA and the FEP plugin are installed in the SCS environment. Signals from the user's environment "U" are portrayed bypassing the FEsP module displayed here on the way to the AWS EC2 instance; on the other hand, the CSS estate is fed by the FEP of the CSP. It is also possible to use this stream to set up an alert that will trigger the sending of a signal in the event of a service level agreement (SLA) violation. For this purpose, SLA monitoring metrics may be added to existing monitoring platforms like AWS CloudWatch, DataDog, and Dynatrace. You may use the data gathered to dig further into service level agreement (SLA) breaches and discover recurring themes. Monitoring systems can send emails or text messages to the right persons or devices if an emergency arises. These notifications may be sent to security incident and event management (SIEM) appliances like Splunk, ArcSight, and QRadar for analysis and further dissemination. Zabbix, RHQ, and fluentd also stand out because of their specialized capabilities for handling alerts from various services.

4.3.8.4. TTP Environment

When confusion or conflict arises over a service, it is the TTP environment (T) that handles the situation. Our deployment method counts on the TTP being installed on a cloud server that also has a FEP module. It also keeps the original signed SLA between the CSP and the CSS. Additionally, it offers a different method for checking and validating communications between these parties to stop attacks like Man-in-the-Middle (MITM) and Man-in-the-Cloud (MITC). We suggest a cluster coordination service that can maintain the consistency of node configuration data as the primary way of establishing the presence of this thing. We see such a service acting as a central node in a distributed environment, holding SLA and other system configuration information. In order to accomplish this, we may use services such as Apache Zookeeper, doozerd, etcd, consul, chef, or puppet to manage and disseminate system configuration and other SLO-related information across all nodes in a distributed system, while also transmitting a periodic yet synchronised convergence.

4.3.8.5. End Users Environment

The user's actual production setting, denoted by the letter U, plays a major part in our implementation strategy. This entity makes use of the cloud services and simulates a complete work environment in order to evaluate the cloud service's performance in light of the Service Level Agreement. In this case, Apache's load testing simulation solution, JMeter, is part of the user's environment, which may be modeled as either Ubuntu or Microsoft Windows. Though there are other systems that do similarly, such as Taurus, SIEGE, LOIC, and BURP, JMeter is useful for running a variety of pre-configured load testing plans and analytic patterns. Its minimal resource usage allows for accurate measurements of the performance and behavior of the intended system, exemplified by the use of a variety of communication protocols as HTTP, SOAP, LDAP, POP3, and SMTP. This study will use an example web application to evaluate AEB environment.

Among other reasons, Jmeter's adaptability has made it the go-to tool for mimicking the user's actual surroundings. This open-source software supports graphical user interface (GUI) and non-GUI projects, runs on several platforms, and can do standalone and networked test runs. Our JMeter test plan contains all the details about the targeted server where the.jar file containing the example web application is located, which is required for the test to execute successfully. Of course, there are other viable options, like hosting the web app in question with an automated application deployment resource like AEB or a comparable SaaS, or deploying it on a standalone application server like Oracle, IBM, VM, Microsoft, or Apache.

Jmeter collects several metrics, including the date, elapsed time, response code, thread name, latency, transaction success, and failure message, when an HTTP request sampler is formed. The RFC4180-specified.csv file extension denotes a comma-separated value.

4.3.9. FEP Implementation Requirements

Certain conditions must first be met before the protocol may be implemented. These factors function independently, but their powerful interplay enables insightful conclusions about the effectiveness of automated SLA enforcement and dispute resolution.

a) A valid SLA between CSP & CSS
b) A User estate
c) A CSP estate
d) A CSS estate
e) A TTP
f) A Message Forwarding Module
g) FE & SLA enforcement modules
h) A safe place to exchange information where there are checks and balances in place to make sure everything checks out

Log analysis and other discovery methods are beyond the focus of this study but may be desirable based on business requirements.

4.3.9.1. Protocol's SLA Negotiation

Providers and customers of cloud services always sign enforceable service level agreements (SLAs). Quality of service (QoS) and related measuring capabilities and metrics, such as latency, for a cloud service must be mutually agreeable between the cloud service provider (CSP) and the cloud service user (CSS) (from high level e.g. availability, bandwidth, average transactions per second, latency to low-level e.g. CPU, memory, etc.). It defines the service's bounds, lays out the limits on its deployment, and describes the potential statuses for the service. QoS monitoring also takes place during "peak" service hours, which are defined as 8:30 am to 6:00 pm, Monday through Friday, with the remainder of the week, including weekends and holidays, being designated "off-peak." Given that the service provider has implemented these safeguards, it is reasonable to conclude that the cloud service is reliable and secure.

Table 4.9: SLA Sample Metrics

SLA Metrics	**Targeted Threshold (peak-time)**
Availability	≥100.00%
Average Response Time	≤50 miliseconds

What constitutes a service failure or SLA violation is determined by comparing the needs of the CSS with those of the CSP.

Explain the meaning of SLA (SLA) In order to test whether or not QoS exceptions arise when these C.U.T.S. entities connect with one another in stateful and stateless, cloud-based SOA systems, we will need a sample SLA to configure our monitors and other sensors. Service Level Agreements (SLAs)-based metrics evaluate both the client and server sides of a transaction to determine how well a service is provided (when start, where start, when end, where end). Despite the availability of several cloud measures, we settled on an ART for use in evaluating the SLA's effectiveness. We acknowledge that there is a basic flow control mechanism imposed on TCP-based traffic, and that other operational factors may have a substantial impact on achieving the desired SLOs in a distributed computing system, such as cloud computing.

In service provisioning in practice, latency is the time it takes from when a client node sends a request until the serving node completes the request or the provided set of tasks. For the sake of this discussion, "service performance monitoring" means verifying that the service is up to par with the SLA. Possible factors affecting response time include low bandwidth, an overabundance of users, a longer processing time, and a more complicated request. As response times get longer, the possibility of breaking a SLA grows. Our Service Level Agreement (SLA) specifies a threshold of 50 ms (milliseconds) for a normal response time over the internet. An ART may be calculated

by dividing the total number of requests generated by JM by the sum of the timeouts that occurred while those requests were being processed.[91]
ART = total violation / total requests x 100

4.3.9.2. Protocol’s Exchange Signal

The SLA of the theoretical framework is presumed to have been signed by CSP "C" and CSS "S." TTP has been furnished with a copy of the same SLA by each of these entities. TTP will retain a copy of this SLA for its own records and will use it as a reference in the case of a dispute over a violation of this agreement.Customers will still be using the cloud service, but the C will still regularly transmit an alert to the S, and the FEP module will evaluate SLA compliance by comparing the actual response time to the target response time specified in the SLA (ART). Additional information, such as VCSS (resp. VCSP), may be bundled with SLA S during transmission with the help of Jmeter.

4.3.9.3. Protocol’s Dispute Signal

After C, S, and T have sent some test transmissions and received acknowledgements in response, the protocol implementation is ready to begin the BAU exchange cycle. It is assumed here that S has already deposited its ICSS to the T as a token, to be redeemed upon C's successful and SLA-compliant service delivery. And so as to avoid massive data manipulation, we only use 15 minutes of samples from these BAU signal transmissions to portray the real environment. When determining whether or not a service has been delivered in accordance with the Service Level Agreement (SLA), we check for a status-code of 200, which indicates that a simple request has been successful within a qualifying period regardless of the method used, such as the GET method of HTTP (Hyper Text Transfer Protocol) executed via JMeter, by receiving our expected responses within the specified SLA time, for example 50 milliseconds.

4.3.9.4. Protocol’s Abort Signal

For the reasons stated above, JMeter will classify any occurrence in which the server returns an HTTP status-code of 4xx bad request, unauthorized, forbidden, or not found;

or 5xx server faults, service unavailable, gateway time out, etc. as a service failure or SLA violation. In such a case, the FEP module will evaluate the situation. As soon as S is able to, she will notify the TTP of the pregnancy termination. To further define the situation, we use Jmeter's duration assertion, the results of which may be either FALSE or TRUE. In this case, one of our shown triggers sent an abort to the entity T based on a periodic monitoring basis, after computing the ART of each unsuccessful thread. This demonstrates service delivery failure or non-compliance with SLAs due to excessively lengthy ART answers (50 ms).

4.3.9.5. Protocol's Resolve Signal

When and how these parties resolve the dispute: suppose our FEP and security modules feed the TTP, who utilizes the actual SLA metrics and master copy of the SLA to decide on a fair settlement, taking into consideration the victimized business. This will be done after the end-of-term review and evaluation of service metrics. Because the dishonest party would be helpless to defend such a condition, the impartial party would be awarded the benefit of dispute resolution.

A brief discussion of how the suggested protocol would launch the resolved signal and how it would configure the TTP and the participants to recognize it through a few repeats is adequate. Take a look at Table 4.2, Rows 4 and 6, and Table 4.8, Rows 11 and 12.

4.3.9..6. Post Reconciliation & Service Restoration

The FEP modules and the SLE(E) would be put into service once all pending disputes have been resolved and all truthful participants have confirmed that they have received their full service credits or financial dues. We will take into account any possible adjustments to the SLA/SLOs to ensure that the SLA metrics are accurate. The service monitoring look-ups would be updated to incorporate any bad QoS statistics previously connected with the causation of any SLA breaches in order to avoid a repetition of the same SLA violations. The sensors keeping an eye out for these problems would be tweaked, and the relevant information would be cascaded to the concerned TAs and the

TTP, to prevent false positives that could lead to an abnormal service termination or unsolicited configuration mismatch at either the CSP or the CSS FEP modules.

4.3.10. Results

This section discusses the outcomes of implementing our approach. About six distinct strategies were evaluated, representing various real-world business situations. These strategies ranged from local host on-premise to distributed computing models like AWS Amazon. The primary goal of these optimization strategies was to investigate the performance of our proposed protocol (with respect to service automation and SLA enforcement) across a variety of computer environments. In order to measure and track different replies while keeping an eye on the QoS factor of our example web app, we choose to use ART as a representative SLO metric for Quality of Service. Knowing where to put monitoring sensors has always been a difficult problem, thus we kept our MF separate, outside the end user trust boundary, on a specialized appliance system. It would help to get better, more precise findings. Initially, an MF's sole responsibility was to reroute communications, but subsequently, their duties were expanded to include serving as a sensor for keeping an eye on various parameters. It records data about transactions and then delivers the information at the predetermined intervals. It generates a comma-separated values report once the monitoring is complete. csv file, which is widely used in many contexts. The definition of the SLO metric threshold is an important step in establishing a practical response time reference. To better match the consistency of the service and other QoS factors, a shorter response time would be preferable. These numbers represent the translation of a response pattern from a CSP to a service endpoint in a decentralized system. In light of our research on the average response time of different web services, we settled on 50 milliseconds as the minimum need for an ART to be SLA compliant. If the response time for any given message exchange is more than 50 milliseconds, the service level agreement (SLA) is considered to have been broken. In this short analysis, we will look at how the SLA measurements we have collected may be immediately applied to four distinct jobs serving as forms of varied reconciliation: [92]

a) Periodic SLA violations

b) Response Time variation

c) Net Message Exchanges

d) TTP‘s Involvement

4.3.10.1. Reviewing Periodic SLA Violations

Having complete, unambiguous insight into the time and circumstances of any SLA breaches is the first and most important factor in drawing conclusions from the data. Here, we synthesize findings from several experiments run in the cloud to demonstrate the efficacy of our cloud-based solution, such as Azure. As shown in Fig. 5.3, we structured the deployment method in a way that allowed us to zero in on the benchmarking of individual service nodes during message exchange. Here, we'll be concentrating on one of our key SLO metrics: average response time (ART). Both an Ubuntu-based application server configured manually and an automated application deployment platform hosted on Microsoft Azure's Elastic Web Services were used in the deployment's testing. The aforementioned deployment underwent a number of tests before being probed against the real transactions over the course of three days, allowing for a more nuanced capture of these outcomes. By using this route, we were able to ensure the consistency, precision, and accurate capture of client/server answers during the testing regime's real-time phase. One other goal was to establish a standard for measuring SLOs that would be representative of actual business practice. One important point concerning the sensor's location is needed to be made now, since it affects the interpretation of these data. The MF node was set up as a separate cloud entity so that it could do many tasks at once, such as forwarding messages to and from other nodes and recording timestamps so that they may be used as evidence in the event of a legal hold.[93]

Table 4.10: CSP‘s Captured SLA Metrics

SLA Metrics	Average Response Time(ms)	Agreed SLA(ms)
SLA Metrics1	43.78	50
SLA Metrics2	47.82	50
SLA Metrics3	49.85	50

SLA Metrics4	34.53	50
SLA Metrics5	38.24	50
SLA Metrics6	39.94	50
SLA Metrics7*	68.09	50
SLA Metrics8	39.07	50
SLA Metrics9*	67.29	50
SLA Metrics10	47.35	50
SLA Metrics11	47.58	50
SLA Metrics12	48.06	50
SLA Metrics13	48.43	50
SLA Metrics14	48.19	50
SLA Metrics15*	100.06	50
SLA Metrics16	40.45	50
SLA Metrics17	39.43	50
SLA Metrics18	38.84	50
SLA Metrics19	39.41	50
SLA Metrics20	44.61	50

The CSP and the TTP will eventually provide the CSS with these crucial business statistics on a regularly basis and as needed. The first column of Table 4.10 contains the collected SLA-Metrics for 20 example transactions. Each business's real ART for service delivery is determined at the conclusion of each transaction and displayed in the second column; the third column displays the SLA (50ms) threshold that will be monitored by each FEP module. This 50ms metric is applied to every single message sent between the end-user estate and the CSP. Prior to signing the SLA, it is recommended by to rigorously test monitoring an accepted service request SLO measure, such as response time, against the CSP's SLA claim.

About 20 separate transactions were logged on the CSP infrastructure and are listed in this table. Both the service provider and the service subscriber agree to regularly exchange these samples on a monthly basis to check for SLA compliance, and these probes are limited to CSS metrics. In accordance with BAU Best Current Practice, the TTP, in its capacity as an approved monitoring body, would also get a copy of these documents. Before beginning the message exchange, all of these parties would undertake a mutually agreed-upon verification and validation procedure. We hypothesised that the status quo (BAU) message exchange would persist for some time (e.g. monthly or a longer service contract term). The provided sample is only a small subset of the total message stream that was passed back and forth between the CSP and the CSS's client. The central decision was to create digital evidence that can be used in court and verified by a forensics expert to settle any potential disagreements between the parties.

These totals would also be useful for the auditing and reconciliation tasks carried out by the automated FEP modules preconfigured at each node. While the service is being provided, this brief metric collection would be performed at random intervals only. If the CSP's supplied services are deemed to be malfunctioning or are undergoing an agreed/pre-scheduled maintenance outage, then the affected system states will be exempted. To keep things simple for our prototype, we just documented three days of the full-service month. Not only would daily exchange of these metrics be impractical (time/money), but it would also be seen as an unnecessary added burden. In addition, it would influence service delivery since it would place an unnecessary strain on both ends' computing equipment and human resources. Taking a look at Table 4.10, we see that there are only three red flags (service entries): sla-metric, sla-metrics, and sla-metric15. Each of these records would be flagged as a potential SLA violation for that day/time/transaction cycle because the ART is greater than 50ms. Keep in mind that these 20 results only represent a sample of transactions over a period of 3 days, and that these transactions only capture periodic message exchange.[94]

Similarly, the monitoring process is graphically represented in Figure 4.7, which corresponds to the information in the table above. The FEP module inside CSS monitoring was set to calculate the SLA, however the three entries showed above the SLA threshold line did not meet the SLA's requirements. The computed ART and related log files provide proof of SLA observance.

4.3.10.2. Reviewing Response Time Variation

Here, we'll investigate the potential causes of response-time fluctuations in the cloud. In computing, the time between when a job is sent to a serving node and when it is completed is known as the net reaction time. In our experiment, we track this metric for each individual purchase, but SLA violations are evaluated based solely on the ART over the course of the entire time frame.

Variation in response times is observed in Table 4.11, which is correlated with Figure 4.8. To accommodate CSS's business needs, an increase in the number of users may cause the SLA response time to rise above the estimated value. Therefore, increasing the number of concurrent requests would lengthen the response time. As a result, unless the SLA is modified to account for the new circumstances, the assertion duration will not be met.

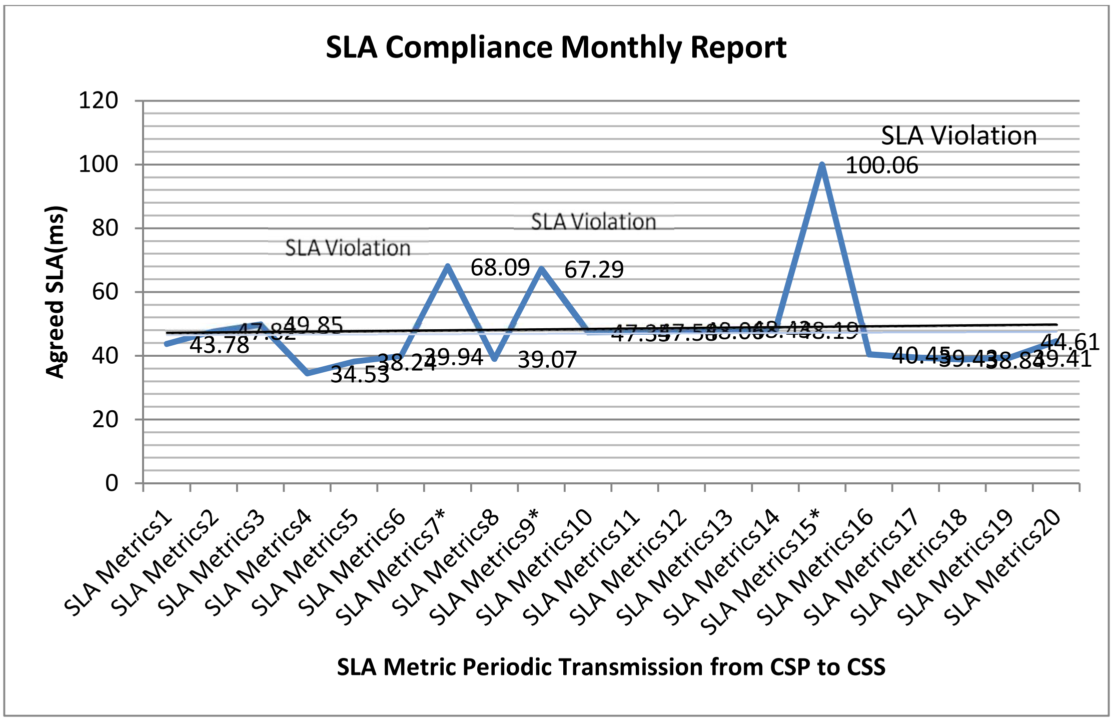


Figure 4.7: SLA Violations & Average Response Times

4.3.10.3.Reviewing Net Message Exchanges

In Fig 5.8, we see a representation of the client's monitoring dashboard for one day of service delivery, complete with the total number of transactions regardless of their success or failure. On days when SLA violations were spotted, a reasonable number of total transactions were processed, but the resulting message exchanges failed to meet the business requirements of providing a timely and polite response. Most of these HTTP requests take longer than usual to respond to. This is a pretty bad Quality of Service sample, in my opinion. This reading follows a service scenario where poor service quality is present, without sacrificing transaction volume. While there is communication, the projected SLA is unacceptable due to financial commitments. Furthermore, we extrapolated from this transmission test a scenario in which we categorise events according to their relative timing. Our SLA monitoring was taken one step further with the introduction of Instance 5.6, which could now keep tabs on the times of day when the service provider saw the highest volume of requests.

Table 4.11: Response Time Variation

SLA Metrics	SLA Violation	Service Variation(ms)
SLA Metrics1	N	6.22
SLA Metrics2	N	2.18

SLA Metrics3	N	0.15
SLA Metrics4	N	15.47
SLA Metrics5	N	11.76
SLA Metrics6	N	10.06
SLA Metrics7*	Y	+18.09↑
SLA Metrics8	N	10.93
SLA Metrics9*	Y	+17.29↑
SLA Metrics10	N	2.65
SLA Metrics11	N	2.42
SLA Metrics12	N	1.94
SLA Metrics13	N	1.57
SLA Metrics14	N	1.81
SLA Metrics15*	Y	+50.06↑
SLA Metrics16	N	9.55
SLA Metrics17	N	10.57
SLA Metrics18	N	11.16
SLA Metrics19	N	10.59
SLA Metrics20	N	5.39

4.3.10.4. Reviewing TTP‘s Involvement

TTP involvement is strictly conditional on anomalous behavior being detected by operational FEP modules. At the outset, our proposed protocol requires businesses to submit to the TTP accurate service metrics that correlate with the actual SLA. This method will help participants make amends in the event of a service dispute in the future.

Concerns raised by TTP during service delivery are investigated in this section of the monitoring process. As can be seen in Fig 5.9, the FEP module of the CSS detects three service anomalies, which are then reported to the FEP module of the TTP. When a dispute arises, TTP notifies the CSP's FEP module so that it can investigate the situation and issue a fair service credit or refund any overpayments.

In this way, the frequency of TTP involvement is shown in Fig 5.9. TTP is used about three times in our transmission tests. After vetting the signals for false positives, the FEP modules at TTP would receive these alerts from CSS and investigate the compromised services. The TTP is programmed to run the following code whenever it receives a signal indicating a violation of the SLA:

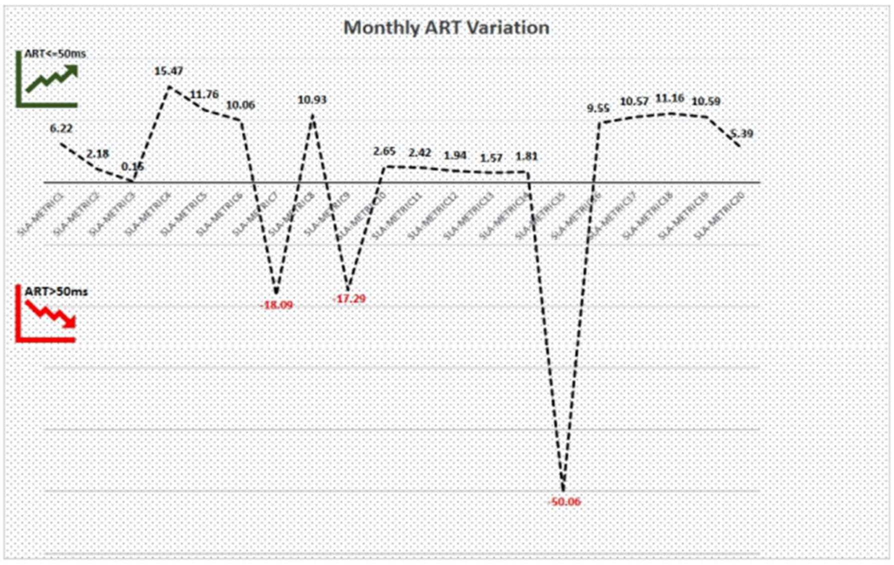


Figure 4.8: Service Response Time Variations (Quazi F.,2020)

```
x >= 50ms
if x >= 50ms  then
print "Warning!!!...SLA violation Detected"\\ or sends an alert via
SIEM (SMS or email) else
print "Transactions are SLA compliant"
```

The above code, located in a module at TTP, would manage transactions in a systematic manner and regulate dispute resolution by labeling messages as either SLA COMPLIANT or NONCOMPLIANT.

In this chapter, we laid out a plan for following through on service level agreements. To show how the prototype performs under these assumptions and with a variety of operating and configuration parameters, a limited implementation was carried out. As so, it demonstrates the success of our quite simple and limited implementation. The test bed also provides access to a number of promising new avenues that will be useful for broadening the implementation's initial scope by making use of flexible new operating assumptions. Such adjustments will undoubtedly enhance the protocol's capabilities, allowing for better and more succinct results in achieving automated SLA enforcement.

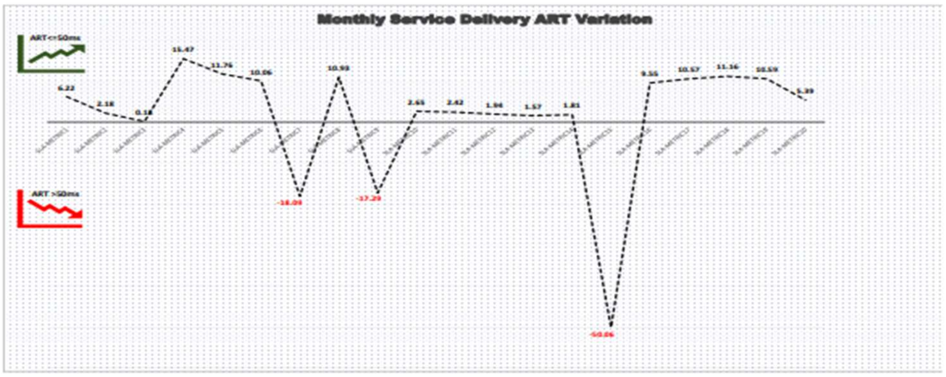


Figure 4.9: SLA Violations & Average Response Times (Quazi F.,2020)

Understanding how to implement in a cloud environment and how to trust and connect underlying technologies is also explored, along with the importance of data security, privacy, and automation. This involves establishing links between various APIs and services to simulate the protocol's behavior. The extensive cloud deployment that is needed for this is, of course, put aside for the time being. Despite this, we have successfully deployed our setup using a few different cloud environments, yielding some very useful and encouraging results that show how our protocol governs service and SLA management through monitoring and detection, as well as enforcement. In the next chapter, we'll take a look at how other potential protocol variances can be evaluated and modeled when participants' bad intentions are already in play. What we proposed to do to counteract the ways in which their bad behavior might compromise equity once the service contract has expired.[95]

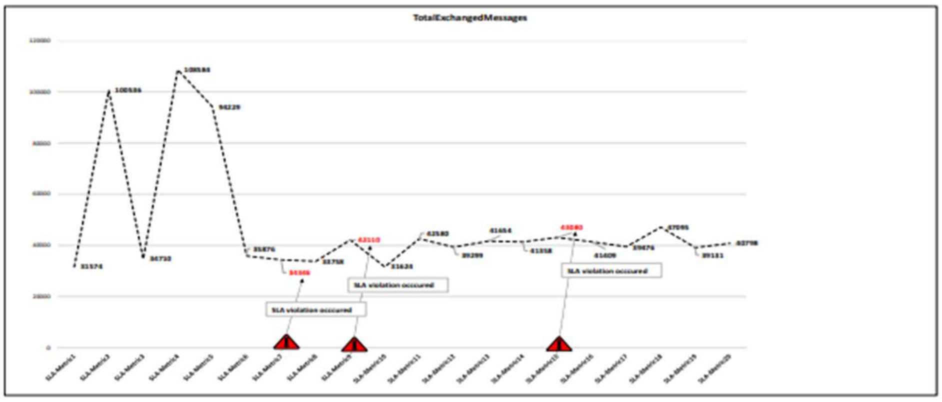


Figure 4.10: Total Exchanged Messages (Quazi F.,2020)

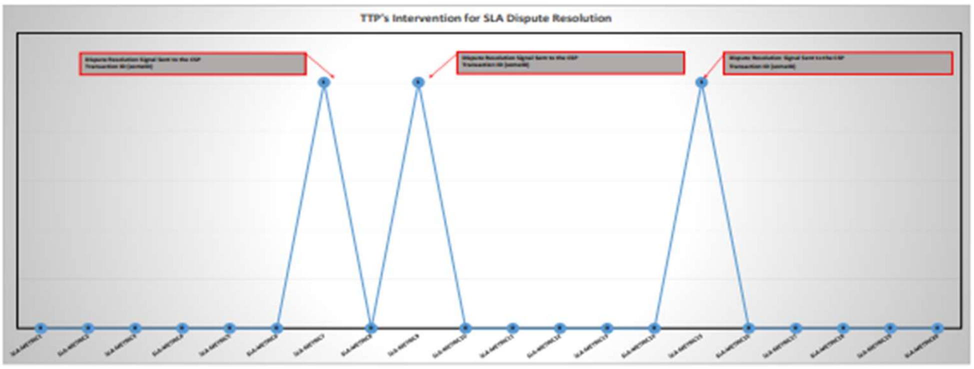


Figure 4.11: TTP Intervention for Dispute Resolution

4.11. Enforcement of SLA with Malicious Participants

The world's tech industry is spending a lot of money to make sure that most businesses move to the cloud, which means that traditional on-site data centres will be phased out and replaced by a broader cloud computing infrastructure from which businesses and consumers can access a wide range of on-demand computing services. Still, a large majority of businesses, of all sizes and in all industries, are taking advantage of the many advantages offered by multi-tiered cloud services, which host everything from basic web apps to complex distributed networks that link businesses together using the latest and greatest in technology. With the advent of the global pandemic, telecommuting has become the norm. According to Gartner's trend, an increasing number of companies throughout the world are drawn to move their services to cloud environments since doing so eliminates the problems associated with remote labour while dramatically lowering the companies' legal risks. As stated in the paper, by 2020, public cloud services are expected to have grown by 6.3%. SaaS, among other cloud services, is still in high demand since it helps businesses adopt online office automation software suits through the payment of online subscriptions, which reduces the need for software licences, maintenance costs, and security risks.

The above computing developments are exciting, but they also present a number of new hurdles for the businesses that are signing up to take use of them in the cloud. The significance of SLAs was previously discussed in Chapter 2. Although SLAs, as the primary legal contract between a CSP and the CSS, do hold both parties accountable and give them some measure of protection, there are still many associated challenges and security issues that could be exploited if one party to the contract decided to behave badly or unfairly toward the other. Sometimes people act in such ways without intending to do harm to others. It might be the result of faulty equipment or a lapse in procedure. Rather than taking unresolved disagreements to court, parties should first seek out SLA bindings, which include information on SLA compliance, violation, accountability, dispute settlement, and claim lodging (service credits/financial reimbursements).

In Chapter 5, we looked at a scenario in which loss-averse participants might benefit from an automated SLA enforcement procedure. Now the environment represented in Fig.6.3 might be applicable if another situation is addressed in which the entities involved display unusual behavior. For instance, the CSS may request a refund if it determines that the CSP is not providing the promised digital goods or services. The onus of proof for the accusations made by the CSSs has shifted since their arousal of the issue. In a similar vein, the CSP claims that CSS has not paid for the services it has received. This is one way in which any party to a service agreement might intentionally harm the other during the duration of service. Since the CSS won't be able to collect low-level service metrics from the CSP's surroundings, a mutual resolution is improbable. If the situation progresses to the judicial system, the complainant will likely be the one with the burden of evidence. In today's legal systems, parties are encouraged to explore all possible avenues for amicably resolving any outstanding service complaints before taking the matter to court. As was previously said, mediators and arbitrators can help with compensation and renegotiation for any contested SLA infractions.

Indeed, a complaint may have to invest a great deal of time and money into a legal proceeding. Trying out some automatic solutions to problems with SLAs is strongly recommended. Because of this, no one involved can possibly find fault with the method used to settle any disagreements. The prototype shown in Chapter 5 is vulnerable to being hacked by malevolent users. As In order to assist CSS balance their usage and, more crucially, determine if the SLA has been broken or not, CSP provides monthly SLA summary reports. Now, however, the question arises as to how to allay the concern that the CSP may alter the data in the file containing the summary of the SLA that is sent on a monthly basis to the CSS before sending it on. Their goal could be to trick the CSS into thinking the SLA is doing well while in reality it isn't. If you made changes like that, CSS wouldn't notice.[96]

This digital service trade platform may become a hostile environment necessitating heightened security and privacy safeguards if other prospective dangers are allowed to compound the existing difficulties. Older designs can only function in an exchange setting if loss aversion is managed by an in-line TTP. If the operating environment were to shift, the protocol would need to be updated to deal with the newer and more dangerous threats and hostile actors.

Based on what we learn about the protocol's features in Chapter 5, we may deduce that it will primarily focus on poor fairness. Due to the characteristics of weak fairness, such as the need to collect a sufficient piece of (forensically sound) evidence utilising various security apertures (such as secure coprocessors, completely homomorphic encryption modules, etc.), this is not always possible. An additional benefit of these kinds of agreements is that they may be used to gather indisputable evidence and provide a solid dispute resolution mechanism. The primary attributes that fulfil the CIA triangle of confidentiality, integrity, and availability are non-repudiation of origin and non-repudiation of reception.

Architecture Threat Model

In order to assess the vulnerabilities in the security of a system's architecture or a related process, threat modelling provides a systematic technique. It uncovers the weak points in a computer, service, or process that an attacker or bad business participant might exploit or misbehave to circumvent security and fairness measures. Since a threat is only a warning about an unprotected (system or design) flaw(s) with the potential to cause harm, threat modelling is the method used to map these hidden dangers so that appropriate preventative actions may be taken. When examining the security of a project's implementation, threat modelling may help teams account for unforeseen events that might disrupt services if a hostile actor (whether human or machine) intervened at any point in the process and used any available exploit. A potentially exploitable entity is one that provides a service, an object that provides a service, a transaction component, or even a supporting entity. In the long run, this might result in full control of the compromised system or service.[97]

We previously introduced a unique prototype with a running environment that provides evidence of a service exchange amongst prime players who are notoriously risk-averse. Using a new operational dimension and a proactive strategy, we simulate potential hostile situations by "war gaming" the protocol. This will provide us the opportunity to talk about the ways in which a participant has been malevolent and how we could protect ourselves. We need to know what would happen if any participant in the protocol decided to behave maliciously while it was being performed, and how our protocol would react to that circumstance to determine who would assume control and make any necessary adjustments. As an example, please describe the outcomes that PA would attain following the conclusion of the agreed cycle. We may get a sense of the gravity of the problem if we suppose that the protocol will die in one of four possible futures, which we will refer to as S.T.E.P.

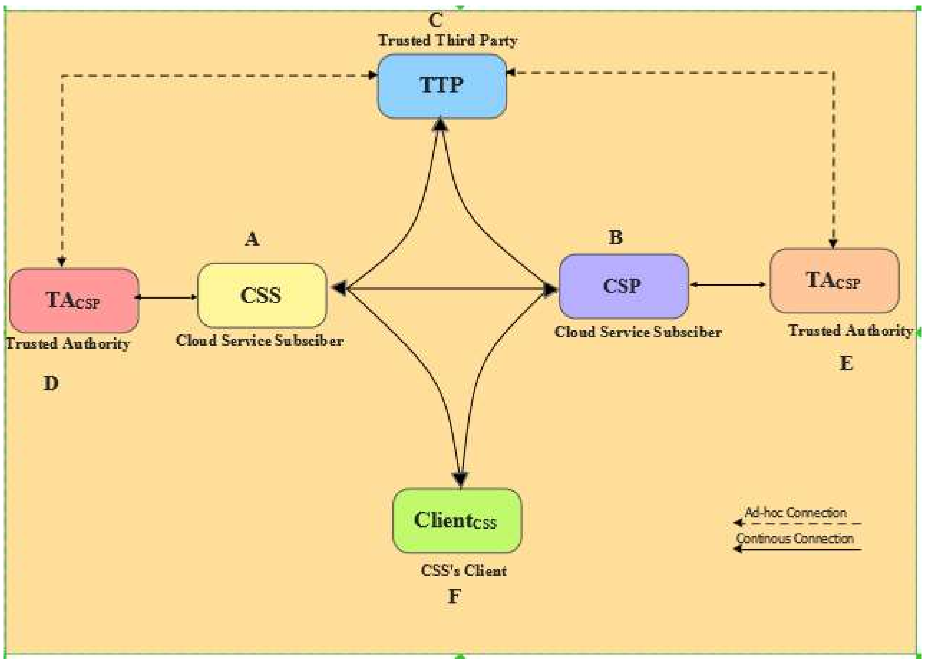


Figure 4.12: Cloud Service Exchange Basic Model

- If St(1,1) has both MB and AckB, the exchange was successful (A). Possessing both KA and KB would also make it simpler to derive IB from MB.
- If a node is in state SA(1,0), it has received MB but no AckB(A) and has requested that the TTP settle the exchange by sending message ResA containing both MA and MB, it is in state St(1,0).
- St(0,0) indicates that no acknowledgments have been received,
- St(0,1) indicates that just AckB(A) has been received. By sending the TTP the message ReqA with MA, PA asks to cancel the conversation.

Basic Service Exchange Model - Potential Threats

As shown in the illustration, a general cloud service exchange model involves the systematic interaction of six different transaction actors. When necessary or in response to instructions from TTP, both the CSP and the CSS will rely on trusted authorities (TAs) to verify and guarantee on their behalf.[98]

Table 4.12: Participants Interaction during Basic Service Exchange/round completion

Steps	Participant Sends	Participants Receives	Message Exchanged	Fairness	TTP Inc.	TA Inc.
1	A	B	Payment	N	N	N
2	B	A	Ack	N	N	N
3	A,B	TTP	Ack(TTP)	N	Y	N
4	TTP	A,B	Ack(SLA)	N	Y	N
5	B	A	CloudService	N	N	N
6	B	A	SLASummary	N	N	N
7	A	B	ServiceTerminates	N	N	N
8	B	A	ServiceTerminates	N	N	N

Covering CSPs as their TA is a certificate authority (CA), while a financial clearinghouse (such as a bank) stands in for CSS and provides payment guarantees on their behalf; TTP mediates any disagreements that may arise. From what can be gleaned from a careful examination of the service exchange model, there aren't that many threats on either side of the communication channels or inside the associated trust limits. Since both parties must perceive a conflict resolution procedure based on manually aided (using trusted persons) or completely automatic (using some smart secure technologies) structures, it is imperative to advocate for the best solutions that can reduce such hazards.[99]

Comparative analysis execution time lock and unlock process CryptomatorTEA and AES application

The findings of a time study on the procedure used to lock and unlock a vault Tables and charts contrasting the TEA implementation of Cryptomator with the AES implementation of Cryptomator are provided. Time spent doing comparisons is seen in Table 4.15.

Table 4.13 Percentage of comparison Execution Time Lock

File Type	File Size	Execution Time Lock TEA(ms)	Execution Time Lock AES(ms)	Percentage of Comparison
.txt	1330	37.92	38.52	1.56
.c	23900	39.87	44.64	10.68
.java	18370	38.43	41.76	7.97
.xlsx	56560	41.46	43.56	4.82
.docx	1630000	158.86	171.18	7.19
.pdf	2150000	254.16	303.25	6.188
.pptx	2410000	215.81	403.87	46.56

It can be seen from comparing the percentages of execution time and lock vault in the TEA Cryptomator and the AES Cryptomator applications that the range of execution time for the lock vault process varies from 1.56 percent to 46.56 percent. The comparison execution time graph is shown in Figure 4.13, the comparison execution time unlock procedure is shown in Table 4.16, and the comparison execution time unlock graph is shown in Figure 4.14.

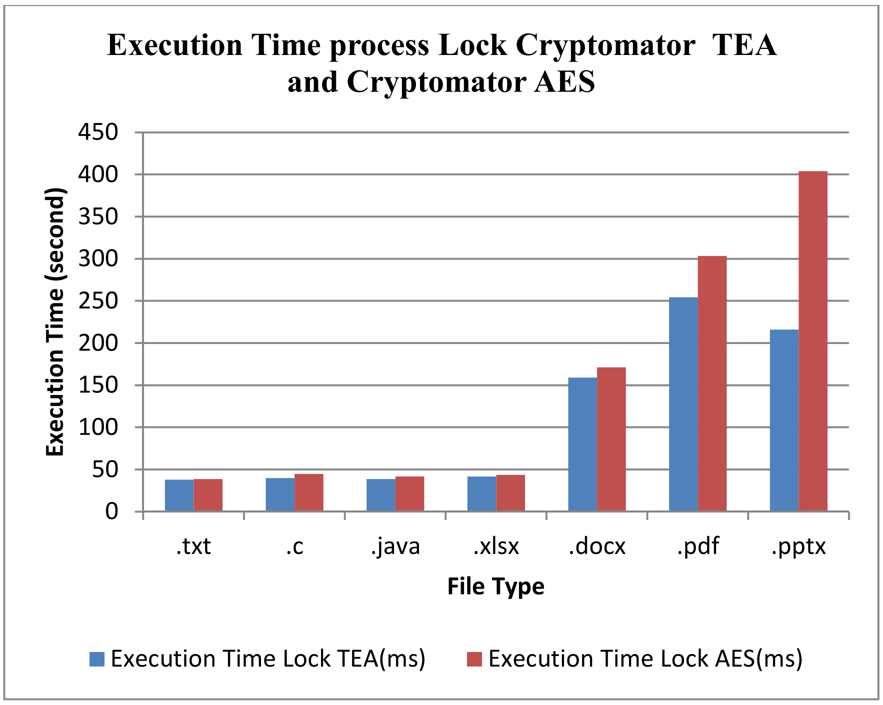
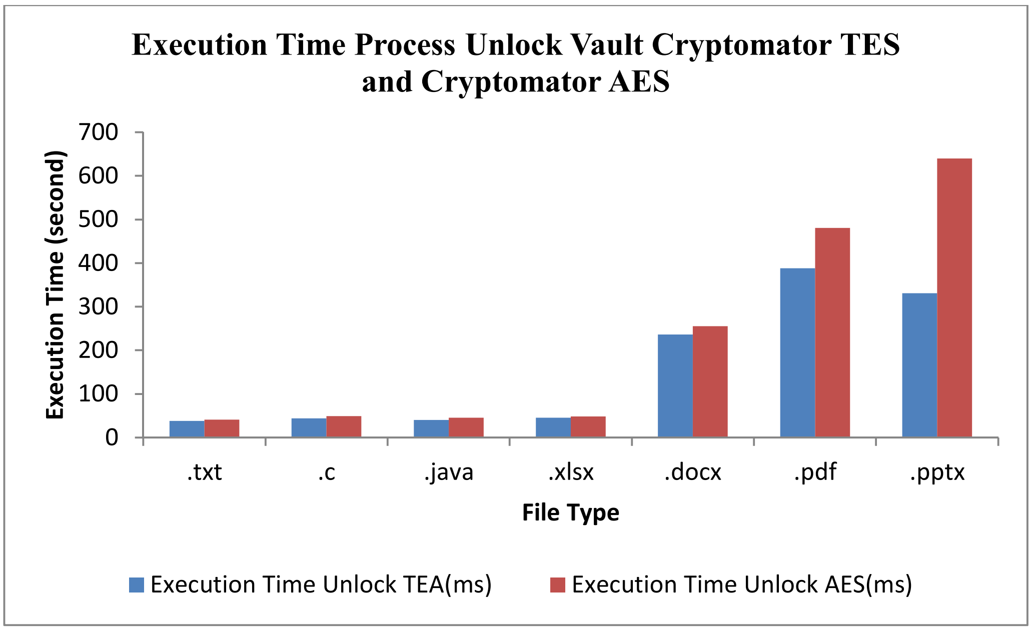


Figure 4.13. Graph Comparison Execution Time

Table 4.14. Comparison Execution Time Unlock

File Type	File Size	Execution Time Unlock TEA(ms)	Execution Time Unlock AES(ms)	Percentage of Comparision(%)
.txt	1330	38.03	40.83	6.86
.c	23900	44.02	49.14	10.42
.java	18370	40.69	45.49	10.55
.xlsx	56560	45.33	48.41	6.36
.docx	1630000	235.98	255.65	7.69
.pdf	2150000	388.17	480.95	19.29
.pptx	2410000	330.75	639.68	48.29

Figure 4.14. Graph Comparison Execution Time Proses Unlock



The time it takes to lock and unlock a vault has certain characteristics, as seen in a graph comparing the two operations across seven standard file types. There is a clear correlation between file size and the amount of time required to analyze its properties. When comparing files of different sizes and data complexity, the execution time increase is negligible for tiny files like c, java, txt, or xlsx, but considerable for huge files like PDF, DOCX, and PPTX.

Comparative analysis performance throughput lock and unlock process Cryptomator TEA and AES application

Information gathered from monitoring the efficiency with which a safe is locked and unlocked the differences and similarities between the Cryptomator TEA program and AES Cryptomator are laid forth in the form of tables and graphs. Throughput comparisons between TEA and AES are provided in Table 4.15 for the locked state, and in Table 4.16 for the unlocked state.

Table 4.15 Comparison Throughput Lock Cryptomator TEA and AES

File Type	File Size	Throughput Lock TEA(b/s)	Throughput Lock AES(b/s)	Percentage of Comparison
.txt	1330	35842.42	34827.81	2.91
.c	23900	606423.53	540880.1	12.12
.java	18370	484730.64	442640.3	9.51
.xlsx	56560	1365319.55	1299422	5.07
.docx	1630000	10295396.19	9548802	7.82
.pdf	2150000	8472615.43	7102390	19.29
.pptx	2410000	11189340.63	5987292	86.88

Table 4.16 Comparison Throughput Unlock Cryptomator TEA and AES

File Type	File Size	Throughput Unlock TEA(ms)	Throughput Unlock AES(ms)	Percentage of Comparison
.txt	1330	35348.39	33192.11	6.191
.c	23900	489683.34	490135.9	0.0837
.java	18370	456396.54	405994.4	11.38
.xlsx	56560	1257220.41	1180799	5.88
.docx	1630000	6914168.56	6388047	5.51
.pdf	2150000	5545276.15	4509357	14.58
.pptx	2410000	7300700.72	3772457	58.93

The theoretical study of the chosen algorithms was based on a literature survey by many scholars. When memory usages, output bytes, and battery life are critical concerns for communication security, encryption techniques play a crucial role. For this purpose, we employ DES, 3DES, AES, and Blowfish as our chosen algorithms.

Table 4.17 Comparative Analysis of Symmetric Encryption Algorithms

Features	DES	3DES	AES	Blowfish	References
Created By	IBM in 1975	IBM in 1978	Joan Daeman, Vincet Rijmen in 1998	Bruce Schneier in 1998	Stallings[17], Forouzan[7] , Schneier[13]
Algorithm Structure	Feistel Network	Feistel Network	Substitution, Permutation Network	Feistel Network	Stallings[17], Schneier[13]
Block Size	64 bit	64 bit	128 bit	64 bit	Stallings[17], Forouzan[7] ,
Rounds	16	48	10,12,14	16	Stallings[17], Schneier[13]
Key Length	56 bits	112,168 bits	128,192 or 256 bits	32 bits to 448 bits	Stallings[17], Forouzan[7] , Agrawal et al.[2], technet[25]
Computational Speed	Fast	Moderate	Fast	Very Fast	Jeeva et al.[8] Agrawal et al.[2]
Tunability	No	No	No	Yes	Jeeva et al.[8]
Encryption Throughput	Medium	Low	High	Very High	Seth et al.[14], Alam et al.[9]
Decryption Throughput	Medium	Low	High	Very High	Seth et al.[14], Alam et al.[9]
Power Consumption	Low	Highest	Medium	Lowest	Marwaha et al.[10], Alam et al.[9]
Memory Usage	High	Very High	Medium	Very Low	Seth et al.[14], Mandal et al.[9]
Security against attacks	Brute force	Brute force,Chosen plain text,known plain text	Chosen plain text,known plain text	Dicitonary Attacks	Jeeva et al.[8] , Agrawal et al.[2] , Cornwell [5]
Cofidentiality	Low	High	High	Very High	Marwaha et al.[10], Cornwell [5]

Table 4.17 shows that the basic algorithmic structure of DES, 3DES, and Blowfish remains unchanged since the early 1970s when it was based on the Fiestel Network established by cryptographer Horst Feistel. But AES used a network based on permutations and substitutions. When encrypting or decrypting data, the smallest possible unit is the block size. For a particular algorithm, a larger Block size increases security but slows down encryption and decryption times. Increased safety is the result of more widespread implementation. A block size of 64 bits has been practically ubiquitous in block cipher construction because it is seen as a suitable compromise. The DES, 3DES, and Blowfish algorithms all require 64-bit blocks. However, AES has a block size of 128. It's safer to use a larger block size. However, the implementation costs (in terms of gates or low-level instructions) of a big block size are higher. The algorithm's number of rounds is a crucial measure of its safety. Having more than one round of defense is a good idea. The crux of the Feistel cipher is that its security is compromised after only one cycle. Both DES and Blowfish have 16 rounds. With 48 rounds, 3DES is three times as secure as DES. However, AES varies depending on the key length; for a 16-byte key, there are 10 rounds, for a 24-byte key there are 12, and for a 32-byte key there are 14. Key management is the fundamental feature that reveals the inner workings of encryption and decryption algorithms. Key search attacks, often known as brute force attacks, may be used to break symmetric key encryption. In these assaults, the attacker systematically attempts every potential key to decode the message until they succeed. In most cases, the target is compromised before all viable keys have been tested. By expanding the potential number of permutations, longer key lengths reduce the likelihood of successful assaults. Key length is adaptable in symmetric algorithms like DES, 3DES, AES, and Blowfish. Blowfish excels because it has the largest key length. The time it takes for an encryption algorithm to transform plain text into encrypted text is referred to as the encryption time. The speed of an encryption system may be estimated by dividing the amount of plaintext in bytes encrypted by the time it took to encrypt the data. In terms of symmetric methods, the research reveals that the Blowfish algorithm requires the least amount of time to encrypt data, while 3DES requires the most time. When compared to DES, AES takes longer to encrypt data. Based on the results of the Encryption Throughput test, it was determined that Blowfish is the most effective and efficient block cipher currently available, surpassing DES, 3DES, and AES. How long it takes for a decryption technique to turn plaintext into ciphertext is what is known as the decryption time. The throughput of a decryption system may be computed by dividing the total number of bytes of ciphertext decrypted by the decryption time. In terms of symmetric algorithms, the research reveals that the Blowfish method requires the least amount of time to decode while the 3DES technique requires the most time. AES takes longer to decrypt than DES. Compared to DES, 3DES, and AES, Blowfish was shown to have superior performance and efficiency in terms of decryption throughput. For portable, battery-operated devices, the encryption algorithm's power consumption is a crucial consideration. In comparison to DES and AES, 3DES is the most power-hungry symmetric-key method. However, compared to DES, 3DES, and AES, Blowfish has the lowest power requirements. We compared Blowfish's power use to that of AES and discovered that it used around 16% less energy. When compared to DES and AES, 3DES is the most memory-intensive symmetric key technique. When compared to DES and 3DES, AES uses less memory. Yet Blowfish makes the fewest memory-intensive requests. Security in cryptography refers to the resistance of an encryption system against brute force and other types of plaintext-cipher text attacks. According to the results, AES is safer than DES and 3DES, two other symmetric algorithms. While DES, 3DES, and AES are all effective block ciphers, Blowfish has a better reputation for security. There was no backdoor vulnerability found in Blowfish, and the key size could not be reduced, hence the conclusion was drawn that it provided long-term data security. DES's lackluster secrecy is a result of its short key length. We conclude that AES may be used in settings requiring stringent safety measures. Blowfish may be employed if high performance is required. When compared to the other algorithms discussed, Blowfish has the highest level of privacy. From Table 4.18, we may infer that Blowfish is more flexible and has a higher encryption/decryption throughput than DES, 3DES, and AES. Blowfish uses far less resources (both power and memory) than the DES, 3DES, and AES algorithms.

Comparison With Other Algorithms

Table 4.18 Comparison between encryption algorithms

S.N.	**Properties**	**Enhanced Tiny Encryption Algorithm**	**Data Encryption Standard (DES)**	**Rivest -Shamir Adleman(RSA)**	**Advanced Encryption Standard (AES)**
1	Key Size	128 bit Key	56 bit Key	1,024 to 4,096 bit typical	128,192 or 256 bits
2	No. of rounds	64 (32 Cycles)	16	1	10,12 or 14 (depending on key size)
3	Block Size	64 bits	64 bits	No Block	128 bits
4	Structure	Feistel network	Balanced Feistel Network	Structure less	Substitution, Permutation Network
5	Advantage	Provides both encryption, embedding and secure data transmission.	Hardware implementations of DES are very fast	The biggest advantage of RSA is that it uses asymmetric keys	Safe Brute Force(128 bit=2128 attempts) Unbreakable as for now.
6	Disadvantage	ETEA suffers from equivalent keys and can be broken using a related key attack requiring 223 chosen plaintexts and a time complexity of 232	DES is now considered insecure because a brute force attack is possible and DES was designed for software and hence runs slowly.	The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasible decode the message	AES-128, the key can be recovered with a computational complexity of 2126.1 using bicliques

CHAPTER 5

5. 0.Conclusion and Future Work

5.1 Conclusion

This study examined the cloud service fabric with a particular emphasis on SLA enforcement with reference to conceivable participant misconduct. The SLA, which serves as the main tool, is surrounded by other tools, approaches, and procedures for compliance. When it comes to SLA monitoring or detection, both open-source and proprietary solutions and frameworks put the most emphasis on the creation of centralized configuration management tools and their accompanying APIs. This is the responsibility of the cloud service provider, leaving the service subscriber with unclear and limited options for sharing true Quality of Service statistics that represent the fundamental parameters of the service being developed. Our proposed framework's architecture unquestionably offers an alternative approach by proposing a major change in cloud service governance and provisioning for both the CSP and the CSS, and by enforcing the SLAs in a truly impartial and fair-centric manner.

During the deployment of cloud services, defining, executing, and upholding security-balanced cloud SLAs is crucial for both CSP and CSS. From the beginning of the service term to the conclusion, SLA enforcement maintains the strategic fairness component. When a contract for one or both CSP and CSS expires, it is very uncommon for the parties to be left with just a few vague data and service reconciliations. Until low-level infrastructure investigations and permitted digital forensics methods are used to evaluate particular and targeted service aspects to verify whether or not service metrics are SLA compliant, a genuine return on investment (RoI) cannot be justified. [97]It is not practical, and the relevant CSP usually never gives their blessing to, the subcontracting of cloud-based services via a plethora of outsourced global brokerage service channels. In this research, we take on this uncertainty in SLA enforcement by framing it as a market efficiency issue. Because it establishes a transparent equilibrium from which SLA enforcement may benefit, the fair exchange protocol is the most reasonable and workable choice.[100]

Subcontracting cloud services and signing their respective SLAs may become problematic if end-to-end security and privacy protection aspects of an entity's data (at rest or in transit) are ignored. Substantial financial losses might occur at any point in the supply of a cloud service if a participant made a bad decision or if a hostile middle actor tried to sabotage service provisioning by breaching inadequate security standards.

By incorporating this novel idea into the architectures we propose in this thesis, as well as the fair exchange protocols and their ancillary protocols, we are able to eliminate multiple opportunities for technical deception and add additional security layers to protect against a wide range of threats posed by unknown malicious attackers during the provisioning of a cloud service. If implemented, these strategies might attempt to steal anything of value from the victim during the transaction. Our framework's design ensures the confidentiality of user information and prevents unauthorized changes. A web service's technical or operational processes may be hindered by such actions. It may affect the intended behavior of services by modifying factors such as resource availability, response time, latency, issue detection, fault tolerance, and problem resolution ratios. In addition, the affected partner's image in the market, technical investments, and production capacity might be harmed by unforeseen service outages and other difficulties.[102]

In contrast to previous methods, this study reaps the benefits of integrating trusted modules such as Secure Coprocessors, which protects data while it is at rest, Fully Homomorphic Encryption, which safeguards data while it is in transit, and other security modules to add an extra layer of defence, making it that much more difficult for an attacker or malicious participant to attempt a service stat, configuration changes, or any kind of data modifications. [97]There will always be security flaws that may be exploited by malicious actors; for example, it is possible for hackers to infiltrate a TMP's firmware utilizing sophisticated offensive tools and tactics and modern, untrusted communication channels to intercept and modify data in transit. The whole field of cyber security can attest to this fact. Since enforcing SLAs is notoriously difficult, ensuring adequate QoS and QoE requires automating SLA enforcement and ensuring that it is correctly implemented on all participating nodes. Distributed services and their constituent parts will be simpler to provide as a result. Our proposed architecture reduces the workload associated with SLA monitoring, detection, and enforcement by centralizing these functions inside a single, dependable framework. Clear cloud service reconciliation metrics are created for forensically-oriented inquiries into any possible service disputes, and the value of each distributed element of the service is quantified.

In light of the rapid growth of autonomous service control systems in the information technology industry. Significant technological changes are also occurring in the implementation of cloud services, with the shift from human-monitored cloud services like QoS controls to autonomous monitoring and control sensors. Although our suggested system is still in its infancy, it demonstrates its potential for SLA enforcement through heightened security layers and sophisticated controls. Future goals include investigating more possible hypotheses by switching time models, trying additional deception and attacking vectors and strategies, and violating SLA enforcement when many parties are employing fair exchange. The disclosure of new and difficult problems by fault models might provide service providers the authority to assert hitherto unheard-of service restrictions that would affect overall fairness. A participant with malicious intent might gain control by using infringement tactics that influence the sub-protocol modules and the protocol. The qualities of the protocol, particularly non-repudiation, speed, and fairness, should be carefully studied to see what technological, operational, regulatory, and legal requirements may be negated. SLA enforcement offers tremendous research opportunities to enhance and expand a variety of use cases that, in the long run, may provide every stakeholder with a degree of cloud service assurance. The deployment of an architecture across several cloud environments may reveal additional limitations relating to TTPs and CAs fault models, such as Byzantine failures.[103]

By focusing on SLA agreements, this research has attempted to reframe the dynamic between service providers and their clients. As a result, the importance of negotiation protocols for both facilitating the sale of services and resolving dependencies throughout value chains has been highlighted. A generic method of creating executable negotiation protocols is considered as a valuable feature in such settings. Implementing protocols as declarative rules was the solution to combining corporate policies with negotiating encounters. This maintained the separation of labour and promoted reusability with little preparation time. The contributions herein differ from past works in that they do not insist that service providers use a certain protocol; instead, they provide a methodical, guided approach that centers on the three core tenets of protocol design—modeling, verification, and implementation. However, the SBNP protocol is a real-world implementation of the approach that has the potential to replace the current, more limited take-it-or-leave-it offerings.

The EU project SLA@SOI had at least four industrial use cases that have proved the protocol and the negotiation platform. Chained negotiation situations in numerous service industries were among them. The work was also used in the EU project Contrail, albeit for a different negotiation and supply situation. This adoption demonstrates the solution's wider usefulness, including for cloud brokerage and other intermediate services. This work's adoption across projects is a reliable sign that the European Commission's investment in SLA management efforts is paying off.

The study of negotiation tactics has made the intricate dynamics that lie at the root of the SLA gap between consumers and suppliers visible. These factors contribute to the reluctance of parties to implement SLA negotiations for automated service procurement. [147]Two methods that are suited for small to large SLA contract spaces are among the contributions made in this field. These advance the practise of tit-for-tat tactics. However, how well their behavioral factors are adjusted determines how well they perform. This study improves the relationship between business utility and cloud-relevant SLA templates as well as the usefulness of existing SLA from the standpoints of customers and providers.[104]

In the tournament-based assessments, preference conflicts were considered in negotiation domains intended to simulate mission-critical, fault-tolerant, retail services or commodities. State-of-the-art methodologies were utilized in these evaluations. The suggested approach algorithms display levels of stability that are considered acceptable. However, the results demonstrate that no one method is the most promising in all negotiation areas, when competing with various utility ideas and opponent techniques. These findings still contribute to the understanding of a challenging field that is actively being researched.

Keeping the approaches general while establishing their actual relevance was a challenge. It became clear, for instance, that a generic protocol development technique cannot be unduly designed and finished as a collection of libraries. As a result, an approach was developed that effectively combined model checking with declarative rules to create negotiation protocols that function as communicating finite state machines. However, a reasonable grasp is required in these areas in order to benefit from the suggested strategy. Using key points and a specific methodology that may be used as an example, the thesis work has attempted to make this process simpler.

Multiple quality elements were addressed by the work on SLA-aware cloud resource management, whose models were officially established or recycled from earlier works. Some of these models may be the subject of different perspectives from other researchers. Although domain-specific application of the provided models can still be further qualified, this does not restrict the validity of this work.[106] Given that service consolidation is a combinatorial optimization issue, one of the advantages of the suggested strategy is the deployment of meta-heuristic algorithms. However, defining the issue necessitates some familiarity with constraint-based search and multi-criteria optimization. One can consider the diversity of programming paradigms or technologies employed to be a constraint. However, the author is of the opinion that even if the suggested procedures fall under specialist areas, they may be successfully learned with appropriate consideration. The utilization of open source, tried-and-true, cost-free, and, whenever possible, industrial-strength tools may make adoption easier. [105]

5.2 Suggestions for Further Work

Our general idea is centered on two scenarios: loss-averse players and malevolent participants that communicate synchronously with one other through outsourced entities like TTP and TAs. Future research will also look at ways to swap these service arrangements using an asynchronous communication model and quantify threat modeling in order to achieve more fair exchange features. The use of cross-platform cloud service functions and associated QoS requirements to enforce SLAs in a more robust manner through enhanced module configuration would fall under this category.

Investigative options may be improved by real-time service negotiations and dispute resolution in a multi-tenancy cloud environment. There is a sincere desire to engage with various service deployments utilizing alternative frameworks in order to understand the limitations of our architectural design. Extending our study into the enforcement of SLAs would also foresee several natural concerns that may propose protocol optimization in the future, empowering justice, privacy, security, autonomy, and trustworthiness. Our protocol may potentially be used with SOA frameworks like ITSM or other standards that provide assurance.

Additionally, it would be crucial and inevitable to modify our proposed architecture in the future to conform to and incorporate cutting-edge security frameworks like COBIT, NIST, Security Orchestration Automation, and Response (SOAR), Cloud Controls Matrix, best practices, Blockchain, and AI-centric platforms. Such clauses will undoubtedly improve small and medium-sized enterprises' ability to gather and share trustworthy SLA metrics for the soundness of their forensics. They can also be easily incorporated into well-known forensics frameworks for the purpose of looking into potential cloud-based unresolved disputes in the event that participants do not agree to such automated arbitrations.[106]

As part of the research into SLA-aware resource management in clouds, Metaheuristics may be contrasted with global search algorithms like Genetic algorithms. This may be used to examine how well algorithms scale in terms of time and memory as the number of decision variables and the size of the state space increase. Another aspect of automated SLA management is SLA implementation. By understanding resource utilization trends and making proactive adjustments to lessen the need for frequent consolidation, this enables the prevention of SLA breaches. Corrective measures are offered through root cause analysis (RCA) methodologies in the event of noncompliance.[107]

References: -

[1] A. Fernandes, L. F. Soares, J. a. V. Gomes, M. M. Freire, and P. R. Inácio, "Security issues in cloud environments: A survey," Int. J. Inf. Secur., vol. 13, no. 2, pp. 113–170, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10207-013-0208-7.

[2] A. Jansen, "Cloud hooks: Security and privacy issues in cloud computing," in 44th Hawaii International Conference on System Sciences (HICSS). IEEE, 2011, pp. 1–10.

[3] A. Silva, A. S. Ferreira, and P. L. Geus, "A methodology for management of cloud computing using security criteria," in 1st Latin American Conference on Cloud Computing and Communications (LatinCloud). Porto Alegre, Brasil: IEEE, "November" 2012, pp. 49–54.

[4] A. Silva, A. S. Ferreira, and P. L. Geus, "A methodology for management of cloud computing using security criteria," in 1st Latin American Conference on Cloud Computing and Communications (LatinCloud). Porto Alegre, Brasil: IEEE, "November" 2012, pp. 49–54.

[5] Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling Smart Cloud Services Through Remote Sensing: An Internet of Everything Enabler," Internet of Things Journal, vol. 1, no. 3, pp. 276–288, 2014.

[6] Adriana L´opez-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. IACR Cryptology ePrint Archive, 2013:94, 2012.

[7] Afshari, M. Mojahed, and R. M. Yusuff, ''Simple additive weighting approach to personnel selection problem,'' Int. J. Innov., Manage. Technol., vol. 1, no. 5, pp. 511–515, 2010.

[8] Ahmed, L. Liu, B. Yuan, M. Trovati, and J. Hardy, ''Context-aware service discovery and selection in decentralized environments,'' in Proc. IEEE Int.

Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Autonomic Secure Comput., Pervasive Intell. Comput., Oct. 2015, pp. 2224–2231

[9] Ajay D Kshemkalyani and Mukesh Singhal. Distributed computing: principles, algorithms, and systems. Cambridge University Press, 2011.

[10] Al-Aqrabi, ‘‘Cloud BI: A multi-party authentication framework for securing business intelligence on the cloud,’’ Ph.D. dissertation, 2016

[11] Alhamad, M., Dillon, T., & Chang, E. (2010, September). Sla-based trust model for cloud computing. In 2010 13th international conference on network-based information systems (pp. 321-324). IEEE.

[12] Al-hamideh, Wijdan. (2022). Evaluation of SLA Quality of Service in Cloud Computing Environment.

[13] Andrzejak, A., Kondo, D., & Yi, S. (2010, August). Decision model for cloud computing under SLA constraints. In 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (pp. 257-266). IEEE.

[14] Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[15] Awad, S. Kadry, B. Lee, and S. Zhang, ‘‘Property based attestation for a secure cloud monitoring system,’’ in Proc. EEE/ACM 7th Int. Conf. Utility Cloud Comput., Dec. 2014, pp. 934–940

[16] Baarslag, K. Fujita, E. H. Gerding, K. Hindriks, T. Ito, N. R. Jennings, C. Jonker, S. Kraus, R. Lin, V. Robu, and C. R. Williams. Evaluating practical negotiating agents: Results and analysis of the 2011 international competition. Artificial Intelligence, vol. 198, pp. 73-103, May 2013.

[17] Bacon, D. Eyers, T. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch, “Information Flow Control for Secure Cloud Computing,” IEEE TNSM SI Cloud Service Management, vol. 11, no. 1, pp. 76–89, 2014.

[18] Bacon, J. Singh, D. Trossen, D.Pavel, A. B. N.Vastardis, K. Yang, S. Pennington, S. Clarke, and G.Jones, “Personal and social communication services for health and lifestyle monitoring,” in Proc. 1st International Conference on Global Health Challenges (Global Health 2012), with IARIA DataSys 2012, Venice, Italy, 2012

[19] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the Art of Virtualization. Technical Report, University of Cambridge. Available online at: www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf

[20] Baset, Salman. (2012). Cloud SLAs: present and future. Operating Systems Review - SIGOPS. 46. 10.1145/2331576.2331586.

[21] Bhattacherjee, B., Abe, N., Goldman, K., Zadrozny, B., Chillakuru, V. R., Del Caprio, M., and Apte, C. (2006). Using Secure Coprocessors for Privacy Preserving Collaborative Data Mining and Analysis. In

[22] Binu, V. & Gangadhar, Nandyala. (2014). A Cloud Computing Service Level Agreement Framework with Negotiation and Secure Monitoring. 2014 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2014. 10.1109/CCEM.2014.7015474.

[23] Blasi, Lorenzo &Brataas, Gunnar & Boniface, Michael & Butler, Joe &D’Andria, Francesco & Drescher, Michel & Jimenez, Ricardo &Krogmann, Klaus &Kousiouris, George & Koller, Bastian &Landi, Giada & Matera, Francesco &Menychtas, Andreas &Oberle, Karsten& Phillips, Stephen & Rea, Luca & Romano, Paolo & Symonds, Michael & Ziegler, Wolfgang. (2013). Cloud Computing Service Level Agreements -- Exploitation of Research Results.

[24] Boneh, D., and Waters, B. (2007). Conjunctive, Subset, and Range Queries on Encrypted Data. In Proceedings of the 4th Conference on Theory of Cryptography (TCC’07), pp. 53-534.

[25] Botta, W. De Donato, V. Persico, and A. Pescapé, ‘‘On the integration of cloud computing and Internet of Things,’’ in Proc. Int. Conf. Future Internet Things Cloud (FiCloud), Aug. 2014, pp. 23–30

[26] Brandic, I., Music, D., Leitner, P., Dustdar, S. (2009). VieSLAF Framework: Enabling Adaptive and Versatile SLA-Management. In Proceedings of the 6th International Workshop on Grid Economics and Business Models (GECON’09), pp. 60-73, August 25-28, 2009, Delft, The Netherlands.

[27] Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater, ‘‘Provably authenticated group Diffie-Hellman key exchange,’’ in Proc. 8th ACM Conf. Comput. Commun. Secur., Nov. 2001, pp. 255–264.

[28] Buyya, C. S. Yeo, and S. Venugopal, ‘‘Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities,’’ in Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun., Vancouver, BC, Canada, Sep. 2008, p. 1.

[29] Buyya, R., Garg, S. K., & Calheiros, R. N. (2011, December). SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In 2011 international conference on cloud and service computing (pp. 1-10). IEEE.

[30] Carlos Muller, Marc Oriol, Xavier Franch, Jordi Marco, Manuel Resinas, Antonio Ruiz-Cortes, and Marc Rodriguez. Comprehensive explanation of sla violations at runtime. IEEE Transactions on Services Computing, 7(2): 168–183, 2014. URL http://idus.us.es/xmlui/bitstream/handle/ 11441/24591/file_1.pdf?sequence=1&isAllowed=y

[31] Cavalcante et al., ‘‘On the interplay of Internet of Things and cloud computing: A systematic mapping study,’’ Comput. Commun., vols. 89–90, pp. 17–33, Sep. 2016.

[32] Cavoukian, A. (2008). Privacy in the Clouds: A White Paper on Privacy and Digital Identity: Implications for the Internet. Available online at: http://www.ipc.on.ca/images/resources/privacyintheclouds.pdf.

[33] Celesti, F. Tusa, M. Villari, and A. Puliafito, ‘‘How to enhance cloud architectures to enable cross-federation,’’ in Proc. IEEE 3rd Int. Conf. Cloud Comput., Jul. 2010, pp. 337–345.

[34] Celesti, F. Tusa, M. Villari, and A. Puliafito, ‘‘Security and cloud computing: Intercloud identity management infrastructure,’’ in Proc. 19th IEEE Int. Workshops Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE), Jun. 2010, pp. 263–265.

[35] Celesti, F. Tusa, M. Villari, and A. Puliafito, ‘‘Three-phase cross-cloud federation model: The cloud SSO authentication,’’ in Proc. 2nd Int. Conf. Adv. Future Internet (AFIN), Jul. 2010, pp. 94–101

[36] Celesti, M. Fazio, M. Giacobbe, A. Puliafito, and M. Villari, ‘‘Characterizing cloud federation in IoT,’’ in Proc. 30th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA), Mar. 2016, pp. 93–98.

[37] Chappel, D. (2008). Introducing the Azure Services Platform. Available online at: http://download.microsoft.com

[38] Charlotte Kotas, Thomas Naughton, and Neena Imam. A comparison of amazon web services and microsoft azure cloud platforms for high performance computing. In 2018 IEEE International Conference on Consumer Electronics (ICCE), pages 1–4. IEEE, 2018.

[39] Chong, F., Carraro, G., and Wolter, R. (2006). Multi-Tenant Data Architecture. Available online at: http://msdn.microsoft.com/en-us/library/aa479086.aspx.

[40] Cloud Computing Security: Making Virtual Machines Cloud Ready. Available online at: http://www.techrepublic.com/whitepapers/cloud-computing-security-making-virtual-machines-cloudready/1728295

[41] Conte de Leon, Daniel & Bhandari, Venkata &Jillepalli, Ananth & Sheldon, F.T.. (2016). Using a Knowledge-based Security Orchestration Tool to Reduce the Risk of Browser Compromise. 10.1109/SSCI.2016.7849910.

[42] Creeger, M. (2009). Cloud Computing: An Overview. ACM Queue- Distributed Computing, Vol 7, Issue 5, p. 2, June 2009. New York: ACM Press.

[43] Csaplar, “Who is adopting the public cloud faster ? north america or europe ?” Aberdeen Group,Tech.Rep.,July 2013. [Online]. Available:http://www.aberdeen.com/AberdeenLibrary/8565/AI-public-cloudadoption.

[44] D. Petcu, ‘‘Multi-Cloud: Expectations and current approaches,’’ in Proc. Int. Workshop Multi-Cloud Appl. Federated Clouds, Apr. 2013, pp. 1–6.

[45] Dadhich, Manish & Rao, Shalendra&Sethy, Surendra & Sharma, Renu. (2021). Determining the Factors Influencing Cloud Computing Implementation in Library Management System (LMS): A High Order PLS-ANN Approach. Library Philosophy and Practice. 2021.

[46] De Marco, Lucia &Ferrucci, Filomena &Kechadi, Tahar. (2015). SLAFM: A Service Level Agreement Formal Model for Cloud Computing. 10.5220/0005451805210528.

[47] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon’s Highly Available Key-Value Store. In Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles (SOSP’07), pp. 205-220, Stevenson, WA, USA, October 2007.

[48] Desisto, R. P., Plummer, D. C., and Smith, D. M. (2008). Tutorial for Understanding the Relationship between Cloud Computing and SaaS. Stamford, CT: Gartner, April 2008.

[49] Dhanjani, “Hacking Lightbulbs: Security Evaluation of the Philips hue Personal Wireless Lighting System,” 2013, accessed: 28th July 2015. [Online]. Available: http://www.dhanjani.com/docs/ Hacking%20Lighbulbs%20Hue%20Dhanjani%202013.pdf

[50] E. D. Nitto, M. D. Penta, A. Gambi, G. Ripa, and M. L. Villani. Negotiation of service level agreements: An architecture and a search-based approach. Springer Berlin Heidelberg, pp. 295-306, 2007

[51] E. P. Leverett, "Quantitatively Assessing and Visualising Industrial System Attack Surfaces," University of Cambridge, MPhil., 2011

[52] Emeakaroha, V. C., Brandic, I., Maurer, M., & Dustdar, S. (2013). Cloud resource provisioning and SLA enforcement via LoM2HiS framework. Concurrency and Computation: Practice and Experience, 25(10), 1462-1481.

[53] Emeakaroha, V., Calheiros, R. N., Netto, M. A. S., Brandic, I., & De Rose, C. A. F. (2010). DeSVi: an architecture for detecting SLA violations in cloud computing infrastructures. In Proceedings of the 2nd International ICST Conference on Cloud Computing (CloudComp 2010), 2010, Espanha..

[54] Emig, C., Brandt, F., Kreuzer, S., and Abeck, S. (2007). Identity as a Service-Towards a ServiceOriented Identity Management Architecture. In Proceedings of the 13th Open European Summer School and IFIP TC6.6 Conference on Dependable and Adaptable Network and Services (EUNICE'07), pp. 1-8, July 2007, Twente, The Netherlands.

[55] Everett, C. (2009). Cloud Computing- A Question of Trust. Computer Fraud & Security, Vol 2009, Issue 6, pp. 5-7 June 2010.

[56] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier, ''A federated multi-cloud PaaS infrastructure,'' in Proc. IEEE 5th Int. Conf. Cloud Comput., Jun. 2012, pp. 392–399

[57] García, A. G., Espert, I. B., & García, V. H. (2014). SLA-driven dynamic cloud resource management. Future Generation Computer Systems, 31, 1-11.

[58] Garg, Radhika & Stiller, Burkhard. (2015). Factors Affecting Cloud Adoption and Their Interrelations. 10.5220/0005412300870094.

[59] Gellman, R. (2009). Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing. World Privacy Forum (WPF) REPORT, February 23, 2009. Available online at: http://www.worldprivacyforum.org/cloudprivacy.html (Accessed on: January 23, 2013).

[60] Golden, B. (2009). Capex vs. Opex: Most People Miss the Point about Cloud Economics.URL:http://www.cio.com/article/484429/Capex_vs._Opex_Most_ People_Miss_the_point_About_Cloud_Econ omic.

[61] H. Zulkernine and P. Martin. An adaptive and intelligent SLA negotiation system for web services. IEEE Transactions on Services Computing, vol. 4, pp. 31-43, January 2011.

[62] Hassan, Wajid & Chou, Te-Shun & Li, Xiaoming& Appiah-Kubi, Patrick & Omar, Tamer. (2019). Latest trends, challenges and Solutions in Security in the era of Cloud Computing and Software Defined Networks. International Journal of Informatics and Communication Technology (IJ-ICT). 8. 162-183. 10.11591/ijict.v8i3.pp162-183.

[63] Heritage, T. (2009). Hosted Informatics: Bringing Cloud Computing Down to Earth with Bottom-Line Benefits for Pharma. Next Generation Pharmaceutical, Issue 17, October 2009.

[64] Huang, A. Ganjali, B. H. Kim, S. Oh, and D. Lie, "The state of public infrastructure-as-a-service cloud security," ACM Comput. Surv., vol. 47, no. 4, pp. 68:1–68:31, Jun. 2015. [Online]. Available: http://doi.acm.org/10.1145/2767181.

[65] Islam, Chadni& Ali Babar, Muhammad & Nepal, Surya. (2019). A Multi-Vocal Review of Security Orchestration. ACM Computing Surveys. 52. 1-45. 10.1145/3305268.

[66] Islam, Chadni& Ali Babar, Muhammad & Nepal, Surya. (2020). Architecture-centric Support for Integrating Security Tools in a Security Orchestration Platform.

[67] Islam, Chadni. (2020). A Multi-Vocal Review of Security Orchestration. 10.1145/3305268".

[68] Itani, W., Kayssi, A., and Chehab, A. (2009). Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures. In Proceedings of the 8th IEEE International Conference on Dependable, Automatic and Secure Computing (DASC'09), pp. 711-716, Chengdu, China, December 2009.

[69] J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed. Wiley Publishing, 2008.

[70] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, ''Internet of Things (IoT): A vision, architectural elements, and future directions,'' Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645–1660, 2013.

[71] J. L. Garcia, H. Ghani, D. Germanus, and N. Suri, "A security metrics framework for the cloud." in SECRYPT, J. Lopez and P. Samarati, Eds.SciTePress, 2011, pp. 245–250. [Online]. Available: http://dblp.unitrier.de/db/conf/secrypt/secrypt2011.html#LunaGGS11

[72] J. Luna, H. Ghani, T. Vateva, and N. Suri, "Quantitative assessment of cloud security level agreements - a case study," in In Proceedings of the International Conference on Security and Cryptography, ser. SECRYPT 2012. SciTePress, 2012, pp. 64–73.

[73] J. Luna, R. Langenberg, and N. Suri, "Benchmarking cloud security level agreements using quantitative policy trees," in Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop, ser.CCSW '12. New York, NY, USA: ACM, 2012, pp. 103–112. Available: http://doi.acm.org/10.1145/2381913.2381932.

[74] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A Gap Analysis of Internet-of-Things Platforms," 2015, Arxiv, arXiv:1502.01181. [Online]. Available: http://arxiv.org/abs/1502.01181

[75] J. Singh and J. Bacon, “On Middleware for Emerging Health Services,” Journal of Internet Services and Applications, vol. 5, no. 6, pp. 1–34, 2014

[76] J. Singh, J. Bacon, J. Crowcroft, A. Madhavapeddy, T. Pasquier, W. K. Hon, and C. Millard, “Regional Clouds: Technical Considerations,” University of Cambridge, Tech. Rep. UCAM-CL-TR-863, 2014, accessed: 28th July 2015. [Online]. Available: http://www.cl.cam.ac. uk/techreports/UCAM-CL-TR-863.pdf

[77] J. Xu, D. Zhang, L. Liu, and X. Li, ‘‘Dynamic authentication for crossrealm SOA-based business processes,’’ IEEE Trans. Services Comput., vol. 5, no. 1, pp. 20–32, Jan./Mar. 2012.

[78] J.-M. Bohli, N. Gruschka, M. Jensen, L. L. Iacono, and N. Marnau, ‘‘Security and privacy-enhancing multicloud architectures,’’ IEEE Trans. Dependable Secure Comput., vol. 10, no. 4, pp. 212–224, Jul./Aug. 2013.

[79] Jegou, Y., Harsh, P., Cascella, R. G., Dudouet, F., & Morin, C. (2012, October). Managing OVF applications under SLA constraints on contrail virtual execution platform. In 2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm) (pp. 399-405). IEEE.

[80] Kanagasabapathi, Kaaviyan& Deepak, S. & Prakash, Parvathy. (2016). A Study on Security Issues in Cloud Computing. 10.1007/978-81-322-2674-1_17.

[81] Khan, Fiaz. (2016). Service Level Agreement in Cloud Computing: A Survey. International Journal of Computer Science and Information Security (IJCSIS). 14. 324-330.

[82] Kozlov, J. Veijalainen, and Y. Ali, “Security and privacy threats in IoT architectures,” in Proc. 7th International Conference on Body Area Networks (BodyNets). ICST, 2012, pp. 256–262.

[83] Leitner, J. Ferner, W. Hummer, and S. Dustdar, ‘‘Data-driven and automated prediction of service level agreement violations in service compositions,’’ Distrib. Parallel Databases, vol. 31, no. 3, pp. 447–470, Sep. 2013.

[84] Lesjak, T. Ruprechter, J. Haid, H. Bock, and E. Brenner, “A Secure Hardware Module and System Concept for Local and Remote Industrial Embedded System Identification,” in Emerging Technology and Factory Automation (ETFA). IEEE, 2014, pp. 1–7

[85] Liu et al., NIST Cloud Computing Reference Architecture, vol. 500, no. 2011. Gaithersburg, MD, USA: NIST, 2011, pp. 1–28.

[86] Luna Garcia, J., Langenberg, R., & Suri, N. (2012, October). Benchmarking cloud security level agreements using quantitative policy trees. In Proceedings of the 2012 ACM Workshop on Cloud computing security workshop (pp. 103-112).

[87] M. Chhetri, J. Lin, S. Goh, J. Yan, J. Zhang, and R. Kowalczyk. A coordinated architecture for the agent-based service level agreement negotiation of web service composition. In Australian Software Engineering Conference, 2006.

[88] M. Hutter and R. Toegl, “A Trusted Platform Module for Near Field Communication,” in International Conference on Systems and Networks Communications (ICSNC). IEEE, 2010, pp. 136–141.

[89] M. Kazim and S. Y. Zhu, ‘‘Virtualization security in cloud computing,’’ in Guide to Security Assurance for Cloud Computing. Springer, 2015

[90] M. Kovatsch, S. Mayer, and B. Ostermaier, “Moving application logic from the firmware to the cloud: Towards the thin server architecture for the Internet of Things,” in Proc. 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS). IEEE, 2012, pp. 751–756.

[91] M. Rak, N. Suri, J. Luna, D. Petcu, V. Casola, and U. Villano, ‘‘Security as a service using an SLA-based approach via SPECS,’’ in Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom), Vol. 2, Dec. 2013, pp. 1–6

[92] M. University, “Service measurement index,” CMU, Moffett Field, CA, United States, Tech. Rep., 2011.

[93] M. Weiser, “Ubiquitous computing,” Computer, vol. 26, no. 10, pp. 71–72, 1993.

[94] Mario Mac´ıas and Jordi Guitart. Sla negotiation and enforcement policies for revenue maximization and client classification in cloud providers. Future Generation Computer Systems, 41:19–31, 2014.

[95] McGrew and E. Rescorla, “Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP),” IETF, 2010.

[96] Mell and T. Grance, “The nist definition of cloud computing,” National Institute of Standards and Technology (NIST), Tech. Rep. 800-145, September 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[97] Mganga, R. P., & Charles, M. (2011). Enhancing Information Security in Cloud Computing Services using SLA Based Metrics.

[98] Mr. Praveen Kaushik Ms. Shaheen Ayyub. An analysis of security attacks on cloud wrtsaas. International Journal of Advancements in Research & Technology, Volume 4, Issue, February 2015.

[99] Mubeen, Saad &Abbaspour, Sara & Papadopoulos, Alessandro &Ashjaei, Mohammad & Pei Breivold, Hongyu& Behnam, Moris. (2017). Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study. IEEE Access. PP. 1-1. 10.1109/ACCESS.2017.2744677.

[100] Muhammad Faheem Mushtaq, Sapiee Jamel, Abdulkadir Hassan Disina, Zahraddeen A Pindar, N Shafinaz Ahmad Shakir, and Mustafa Mat Deris. A survey on the cryptographic encryption algorithms. International Journal of Advanced Computer Science and Applications, 8(11):333–344, 2017.

[101] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, ''Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems,'' in Proc. IEEE 6th Int. Conf. Cloud Comput., Jun./Jul. 2017, pp. 887–894. [

[102] N. M. Gonzalez, C. Miers, F. F. Redigolo, T. C. M. B. Carvalho, M. A. S. Jr., M. Nslund, and M. Pourzandi, "A quantitative analysis of current security concerns and solutions for cloud computing." Journal of Cloud Computing: Advances, Systems and Applications, vol. 11, no. 1, pp. 1– 18, 2012.

[103] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu, ''Policy-sealed data: A new abstraction for building trusted cloud services,'' in Proc. USENIX Secur. Symp., Aug. 2012

[104] Nie, Guihua& E., Xueni& Chen, Donglin. (2012). Research on Service Level Agreement in Cloud Computing. 10.1007/978-3-642-28744-2_5.

[105] P. Middleton, P. Kjeldsen, and J. Tully, Forecast: The Internet of Things, Worldwide. Gartner, 2013.

[106] P. Ohm, "Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization," UCLA Law Review, vol. 57, no. 6, pp. 1701– 1777, Aug. 2010.

[107] Paladi, N., Michalas, A., & Dang, H. V. (2018, April). Towards secure cloud orchestration for multi-cloud deployments. In Proceedings of the 5th Workshop on CrossCloud Infrastructures & Platforms (pp. 1-6).

[108] Pandey, P. (2021). Security attacks in cloud computing.

[109] Petry, A. (2007). Design and Implementation of a Xen-Based Execution Environment. Diploma Thesis, TechnischeUniversitat Kaiserslautern, April 2007.

[110] Phil Muncaster. Global e-commerce fraud to top usd25bn by 2024. UK / EMEA News Reporter ,Infosecurity Magazine. URL https://www.infosecurity-magazine.com/news/ global-ecommerce-fraud-to-top-25/.

[111] Philipp Leitner, Anton Michlmayr, Florian Rosenberg, and SchahramDustdar. Monitoring, prediction and prevention of sla violations in composite services. In IEEE Web Services (ICWS) 2010, pages 369–376. IEEE.

[112] Prakash Kuppuswamy, Rajan John, et al. A novel approach of designing e-commerce authentication scheme using hybrid cryptography based on simple symmetric key and extended linear block cipher algorithm. In 2020 International Conference on Computing and Information Technology (ICCIT-1441), pages 1–6. IEEE, 2020.

[113] Price, M. (2008). The Paradox of Security in Virtual Environments. IEEE Computer, Vol 41, Issue 11, pp. 22-38, November 2008.

[114] Qazi, F. (2020). Automating SLA Enforcement in the Cloud Computing (Doctoral dissertation, University of Warwick).

[115] R. Lin, S. Kraus, T. Baarslag, D. Tykhonov, K. Hindriks, and C. M. Jonker. GENIUS : An integrated environment for supporting the design of generic automated negotiators. Computational Intelligence, vol. 30, no. 1, pp. 48-70, 2014.

[116] R. M. Savola and H. Abie, "Metrics-driven Security Objective Decomposition for an e-Health Application with Adaptive Security Management," in International Workshop on Adaptive Security. ACM, 2013, p. 6.

[117] Raiffa. The Art and Science of Negotiation. Harvard University Press, 1982.

[118] Ravele, K. J., & Mtotywa, M. (2022). Factors influencing quality and performance of cloud computing platforms defined by personal users in South Africa. International Journal of Research in Business and Social Science (2147-4478), 11(7), 78-91.

[119] Ravele, Khathutshelo&Mtotywa, Matolwandile. (2022). Factors influencing quality and performance of cloud computing platforms defined by personal users in South Africa. International Journal of Research in Business and Social Science (2147- 4478). 11. 78-91. 10.20525/ijrbs.v11i7.2027.

[120] S. Bouchenak, G. Chockler, H. Chockler, G. Gheorghe, N. Santos, and A. Shraer, “Verifying cloud services: Present and future,” SIGOPS Operating Systems Review, vol. 47, no. 2, pp. 6–19, July 2013. [Online]. Available: http://doi.acm.org/10.1145/2506164.2506167.

[121] S. Bouchenak, G. Chockler, H. Chockler, G. Gheorghe, N. Santos, and A. Shraer, “Verifying cloud services: Present and future,” SIGOPS Operating Systems Review, vol. 47, no. 2, pp. 6–19, July 2013. [Online]. Available: http://doi.acm.org/10.1145/2506164.2506167.

[122] S. Bouchenak, G. Chockler, H. Chockler, G. Gheorghe, N. Santos, and A. Shraer, ‘‘Verifying cloud services: Present and future,’’ACM SIGOPS Operating Systems Review, 2013

[123] S. Ferreira, “Uma arquitetura para monitoramento de segurançabaseadaemacordos de níveis de serviço para nuvens de infraestrutura,” Dissertação de Mestrado, Instituto de Computação, UniversidadeEstadual de Campinas, UNICAMP, 2013.

[124] S. Jankowski, J. Covello, H. Bellini, J. Ritchie, and D. Costa, The Internet of Things: Making sense of the next mega-trend. Goldman Sachs, 2014

[125] S. L. Keoh, S. Kumar, and H. Tschofenig, “Securing the Internet of Things: A Standardization Perspective,” Internet of Things Journal, vol. 1, no. 3, pp. 265–275, 2014. [20] P. Urien, “LLCPS: A new security framework based on TLS for NFC P2P applications in the Internet of Things,” in Consumer Communications and Networking Conference (CCNC), 2013 IEEE. IEEE, 2013, pp. 845–846.

[126] S. Pearson, ‘‘Privacy, security and trust in cloud computing,’’ in Privacy and Security for Cloud Computing. London, U.K.: Springer, 2013, pp. 3–42

[127] Sahal, Radhya&Khafagy, Mohamed &Omara, Fatma. (2016). A Survey on SLA Management for Cloud Computing and Cloud-Hosted Big Data Analytic Applications. International Journal of Database Theory and Application. 9. 107-118. 10.14257/ijdta.2016.9.4.10.

[128] SAML. Accessed: Aug. 26, 2018. [Online]. Available: https://developers. onelogin.com/saml [16] Kerberos. Accessed: Jul. 16, 2018. [Online]. Available: http://web.mit. edu/kerberos/

[129] Sayginer, C., & Ercan, T. (2020). Understanding determinants of cloud computing adoption using an integrated diffusion of innovation (doi)-technological, organizational and environmental (toe) model. Humanities & Social Sciences Reviews, 8(1), 91-102.

[130] Sayginer, Can &Ercan, Tuncay. (2020). Critical Factors Affecting Cloud Computing Adoption In Turkish Companies With Diffusion Of Innovation Theory. Pamukkale University Journal of Social Sciences Institute. 10.30794/pausbed.750829.

[131] Sen, Jaydip. (2013). Security and Privacy Issues in Cloud Computing. 10.4018/978-1-4666-4514-1.ch001.

[132] Senarathna, Ishan & Wilkin, Carla & Warren, Matthew & Yeoh, William & Salzman, Scott. (2018). Factors That Influence Adoption of Cloud Computing: An Empirical Study of Australian SMEs. Australasian Journal of Information Systems. 22. 10.3127/ajis.v22i0.1603.

[133] Skafi, M. & Yunis, Manal &Zekri, Ahmed. (2019). Factors Affecting Cloud Computing Adoption: A Literature Review Synthesis And Analysis.

[134] Snehal Mumbaikar, Puja Padiya, et al. Web services based on soap and rest principles. International Journal of Scientific and Research Publications, 3(5):1–4, 2013.

[135] Stankov, I. &Datsenka, Rastsislau&Kurbel, K.. (2012). Service level agreement as an instrument to enhance trust in cloud computing - An analysis of infrastructureas-a-service providers. 18th Americas Conference on Information Systems 2012, AMCIS 2012. 5. 3813-3822.

[136] T. Baarslag, K. Hindriks, and C. Jonker. Effective acceptance conditions in real-time automated negotiation. Decision Support Systems, 2013.

[137] T. Baarslag, K. Hindriks, C. Jonker, S. Kraus, and R. Lin. The first automated negotiating agents competition (ANAC 2010). New Trends in AgentBased Complex Automated Negotiations, pp. 113-135, Springer, 2010.

[138] T. Dierks and C. Allen, "The TLS Protocol Version 1.0," IETF, Tech. Rep., 1999

[139] T. Morris, "Trusted Platform Module," in Encyclopedia of Cryptography and Security. Springer, 2011, pp. 1332–1335.

[140] Takabi, Daniel & Joshi, James &Ahn, Gail-Joon. (2011). Security and Privacy Challenges in Cloud Computing Environments. Security & Privacy, IEEE. 8. 24 - 31. 10.1109/MSP.2010.186.

[141] TU Delft. GENIUS platform. http://ii.tudelft.nl/genius. Accessed: [21/11/2015].

[142] Ullah, K. W. (2012). Automated security compliance tool for the cloud (Master's thesis).

[143] Varsha R Mouli and KP Jevitha. Web services attacks and security-a systematic literature review. Procedia Computer Science, 93:870–877, 2016

[144] W. Ulla, "Automated security compliance tool for the cloud," Master, Department of Telematics, Norwegian University of Science and Technology, NTNU, 2012.

[145] Wan Mohd Isa, Wan Abdul Rahim &Suhaimi, Ahmad Iqbal Hakim &Noordin, Nurulhuda& Harun, Afdallyna& Ismail, Juhaida&Teh, Rosshidayu. (2020). Factors influencing cloud computing adoption in higher education institution. Indonesian Journal of Electrical Engineering and Computer Science. 17. 412. 10.11591/ijeecs.v17.i1.pp412-419.

[146] Wenjing Lou and Yuguang Fang. A survey of wireless security in mobile ad hoc networks: challenges and available solutions. In Ad Hoc Wireless Networking, pages 319–364. Springer, 2004

[147] Yaqub, E., Yahyapour, R., Wieder, P., & Lu, K. (2012). A protocol development framework for SLA negotiations in cloud and service computing. In Economics of Grids, Clouds, Systems, and Services: 9th International Conference, GECON 2012, Berlin, Germany, November 27-28, 2012. Proceedings 9 (pp. 1-15). Springer Berlin Heidelberg.

[148] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson, "The Matter of Heartbleed," in Proc. Internet Measurement Conference (IMC). ACM, 2014, pp. 475–488

[149] Zanella, "Internet of Things for Smart Cities," IEEE Internet of Things, vol. 1, no. 1, 2014