PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
الجمهورية الجزائرية الديمقراطية الشعبية
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
وزارة التعليم العالي و البحث العلمي
UNIVERSITY OF Ammar Telidji LAGHOUAT
جامعة عمار ثليجي بالأغواط

كلية العلوم
FACULTY OF SCIENCES
قسم الإعلام الآلي
DEPARTMENT OF COMPUTER SCIENCES

Master's Thesis

Field: Mathematics and Computer Science
Specialty: Computer Science
Option: Networks, Systems and Distributed Applications

By: **Fatima Zahra Zaoui**

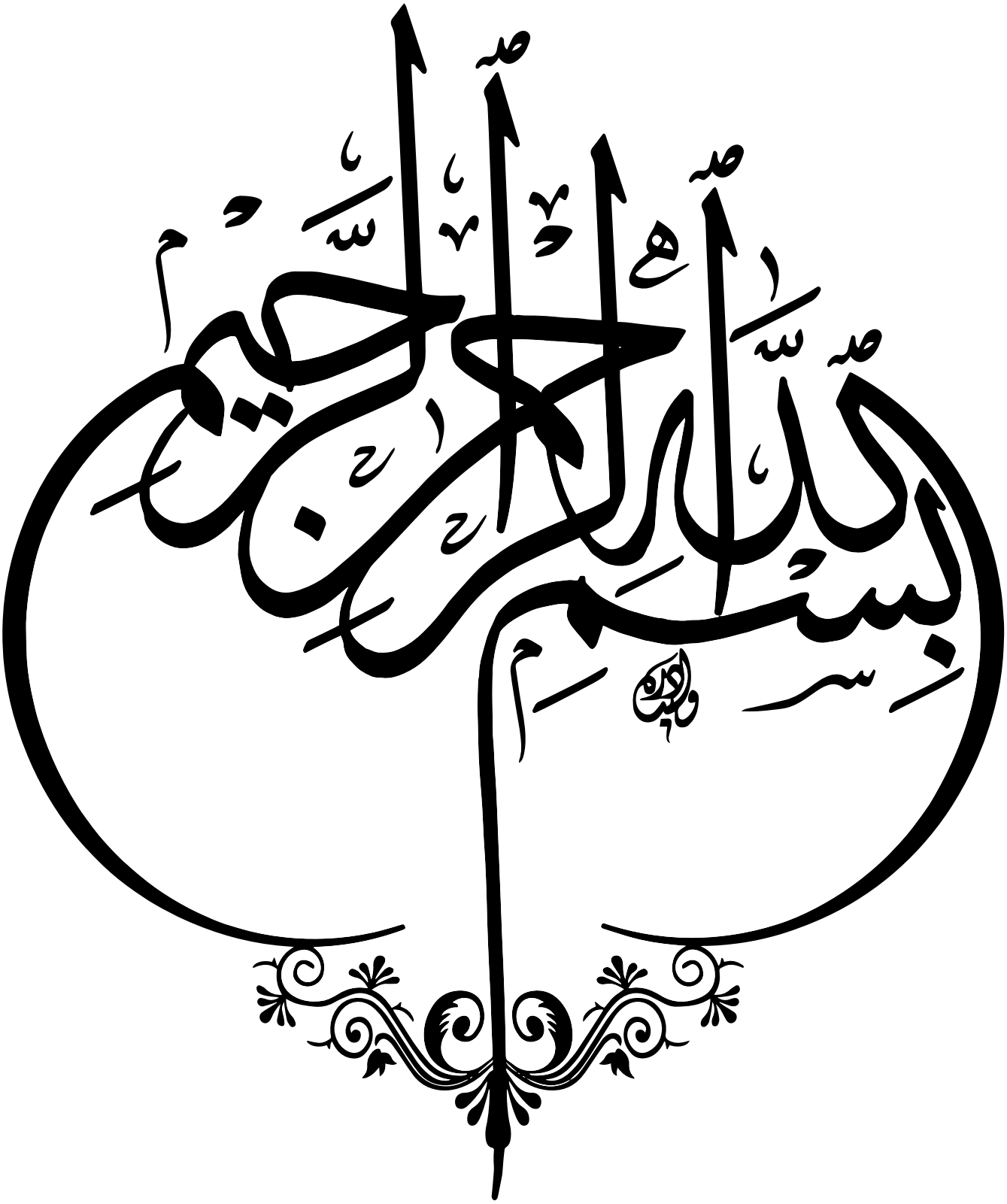
Theme

End-to-End Latency Reduction Techniques for Delay-Sensitive Applications in Satellite-Based Edge Computing Networks

Thesis defended on July 02, 2025 at Laghouat in Front of the Jury Composed of:

Ms. Fatna GUIBADJ	MAA	University of Laghouat	President
Mr. Noureddine CHAIB	Professor	University of Laghouat	Examiner
Mr. Lahcen Mohammed BENSAAD	MCB	University of Laghouat	Examiner
Mr. Messaoud BABAGHAYOU	MAB	University of Laghouat	Supervisor

Academic Year 2024/2025


بسم الله الرحمن الرحيم

Dedication

I proudly dedicate this thesis to all my beloved ones:

- My parents,
- Sister,
- Brothers,
- Especially my two brothers, may their souls rest in peace.
- My best friends.

Acknowledgements

First and foremost, we thank God Almighty. We only began with His guidance, reached our finals with His grace, and achieved our goals with His favor. Praise be to God, who enabled me to take this step in my academic journey. Praise be to God, with thanks, love, and gratitude for the beginning and the end. Our final prayer is to praise God, Lord of the Worlds.

I extend my gratitude towards my supervisor Dr. Messaoud BABAGHAYOU , for his support and guidance and advises, and for his patience and efforts throughout this project, and his expertise were a constant source of inspiration, im very thankful for his help. May God protect and guide you.

I would to offer my sincere gratitude go to the jury members : Dr.Fatna GUIBADJ , Dr.Noureddine CHAIB , and Dr.Lahcen Mohammed BENSAAD , for accepting to review my humble work, and for their valuable feedback, and constructive comments.

To my family, you are my support and strength. Thank you for always being by my side and for your love that gives me strength. May God protect you.

To my friends, who made this journey more enjoyable and less difficult, thank you for every moment of support, every encouraging word, and for all the beautiful memories we made together.

Abstract

With the advent of satellite-edge convergence, which is transforming distributed computing to make services real-time and intelligent, the management of task offloading more efficiently is more important than before. This work addresses the issue of minimizing the end-to-end delay for delay-critical applications in satellite-fueled mist computing systems. This work formulates a new task orchestration algorithm named IsoLink, which was specifically developed and tuned for mist-layer offloading and optimized to classify tasks based on context and assign them wisely to the appropriate virtual machines. IsoLink functions by normalizing the delay and execution parameters, taking into account the type and resource proximity to the task, to enable optimal deployment. By simulating comprehensively using the SatEdgeSim framework, the algorithm performed well in terms of reduced latency, energy savings, and enhanced task success ratio. The evaluation was conducted across various performance metrics and also compared with algorithms such as the Weighte_Greedy, Round_Robin, and Random_VM, and proved the efficacy of IsoLink. Although the method is well-suited for services requiring low latency, future improvements make satellite mist computing environments more adaptive and efficient in operation.

keywords: Satellite Edge Computing, Task Offloading, End-to-End Latency, Mist Layer, Intelligent Orchestration, Delay-Sensitive Applications.

Résumé

Avec l'avènement de la convergence satellite-périphérie, qui transforme l'informatique distribuée pour offrir des services temps réel et intelligents, la gestion plus efficace du déchargement des tâches est plus importante que jamais. Ce travail aborde la question de la minimisation des délais de bout en bout pour les applications critiques dans les systèmes de calcul Mist alimentés par satellite. Ce travail formule un nouvel algorithme d'orchestration des tâches appelé IsoLink, spécifiquement développé et optimisé pour le déchargement de la couche Mist et optimisé pour classer les tâches en fonction du contexte et les affecter judicieusement aux machines virtuelles appropriées. IsoLink fonctionne en normalisant les paramètres de délai et d'exécution, en tenant compte du type et de la proximité des ressources par rapport à la tâche, afin de permettre un déploiement optimal. Grâce à une simulation complète à l'aide du framework SatEdgeSim, l'algorithme a obtenu de bons résultats en termes de réduction de la latence, d'économies d'énergie et d'amélioration du taux de réussite des tâches. L'évaluation a porté sur diverses mesures de performance et a également été comparée à des algorithmes tels que Weight_Greedy, Round_Robin et Random_VM, et a démontré l'efficacité d'IsoLink. Bien que la méthode soit bien adaptée aux services nécessitant une faible latence, les améliorations futures rendent les environnements de calcul de Mist par satellite plus adaptatifs et plus efficaces en fonctionnement.

mots-clés: Informatique en périphérie par satellite, Déchargement des tâches, Latence de bout en bout, Couche de Mist, Orchestration intelligente, Applications sensibles au délai.

مُلخّص

مع ظهور تقارب حافة الأقمار الصناعية، الذي يُحدث نقلة نوعية في الحوسبة الموزعة لجعل الخدمات آنية وذكية، أصبحت إدارة تفريغ المهام بكفاءة أكبر أهمية من ذي قبل. يتناول هذا العمل مسألة تقليل التأخير الشامل للتطبيقات ذات التأخير الحرج في أنظمة الحوسبة الضبابية التي تعمل بالأقمار الصناعية. يُصوغ هذا العمل خوارزمية جديدة لتنسيق المهام تُسمى IsoLink ، طُوّرت خصيصًا لتفريغ طبقة الضباب، ومُحسّنة لتصنيف المهام بناءً على السياق وتعيينها بحكمة إلى الأجهزة الافتراضية المناسبة. تعمل IsoLink عن طريق تطبيع معلمات التأخير والتنفيذ، مع مراعاة نوع المهمة وقرب الموارد منها، لتمكين النشر الأمثل. من خلال المحاكاة الشاملة باستخدام إطار عمل SatEdgeSim ، حققت الخوارزمية أداءً جيدًا من حيث تقليل زمن الوصول، و توفير الطاقة، وتحسين نسبة نجاح المهام. أُجري التقييم باستخدام مقاييس أداء مختلفة، كما قورن بخوارزميات مثل Weight_Greedy و Round_Robin و Random_VM ، وأثبت فعالية IsoLink . على الرغم من أن هذه الطريقة مناسبة للخدمات التي تتطلب زمن وصول منخفضًا، إلا أن التحسينات المستقبلية ستجعل بيئات حوسبة ضباب الأقمار الصناعية أكثر تكيفًا وكفاءة في التشغيل.

الكلمات المفتاحيّة: الحوسبة الطرفية المعتمدة على الأقمار الصناعية، تفريغ المهام، زمن التأخير الشامل، طبقة الضباب، ، التنسيق الذكي، التطبيقات الحساسة للزمن.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

3D Three-Dimensional Coordinate

5G Fifth Generation

6G Sixth Generation

AI Artificial Intelligence

AKG Advanced K-means Algorithm

APOLLO A Proximity-Oriented Low-Layer Orchestration Algorithm

apps applications

AR/VR Augmented Reality / Virtual Reality

BFST Breadth-First-Search-Based Spanning Tree Algorithm

CPU Central Processing Unit

DILIGENT Double-Edge Intelligent Integrated Satellite and Terrestrial Networks

ECS Edge Computing Satellites

ETE End-to-End

FCNs Fog Computing Nodes

IIoT Industrial Internet of Things

IP Internet Protocol

Iaas Infrastructure as a Service

IoT Internet of Things

ISL Inter-Satellite Link

LEO Low Earth Orbit

MEC Mobile Edge Computing

MIPS Million Instructions Per Second

NFV Network Function Virtualization

Paas Platform as a Service

PGRA Potential Game-based Resource Allocation

RAN Radio Access Network

SatEdgeSim Satellite-Edge Computing Simulation Framework

SATIMECN Satellite-Terrestrial Integrated Mobile-Edge Computing Network

SaaS Software as a Service

SEC Satellite Edge Computing

SDN Software-Defined Networking

SMEC Satellite Mobile Edge Computing

STK Satellite Tool Kit

V2X Vehicle-to-Everything

VNF Virtual Network Function

VM Virtual Machine

XML eXtensible Markup Language

Preamble

Cloud, edge, and mist computing have changed the system for delivering computational services over networked environments. Due to the growth of the IoT, billions of smart devices are now producing a large amount of information at the network edge. It has become outdated to trust data processing solely to centralized cloud services, mainly because of the importance of speed and power usage for applications that require constant updates.

This issue was addressed by using edge computing and fog and mist computing, which help to bring processing closer to data sources. Mist computing is gaining attention because it reduces latency by allowing tasks to run on lightweight VMs that are placed close to end-user devices. Besides, satellite network integration with edge computing enables better connectivity and faster processing across the world, most notably in areas with weak infrastructure.

That is why research into ways to intelligently perform offloading on the edge using satellites have recently begun and are the topic of this thesis.

Motivation

With the increasing demand for low-latency applications such as autonomous vehicles, augmented reality, and real-time monitoring, we see a critical need for task-offloading mechanisms that reduce delay and energy consumption. Performing tasks near the data source is a unique platform offered by the mist layer combined with satellite infrastructure. Current offloading strategies often do not consider task type, application profile, and real-time constraints. This very gap has inspired us to design a smart, context-aware algorithm that operates efficiently in mist-level satellite environments.

Problematic

Designing an efficient task offloading strategy for satellite-based mist computing environments presents a challenge that is not trivial. In those environments, tasks come from diverse applications that have their own computational requirements, timing constraints, and resource requirements. Furthermore, such systems are dynamic due to user mobility, varying satellite coverage, changing network latency, as well as the limited availability of virtual machines and energy resources.

The challenge is to issue context-aware and performance-optimized offloading decisions, particularly at the mist layer, where proximity to data sources offers both an advantage and a constraint on processing power and capacity. Moreover, the level of existing algorithms appears to fail in providing a real-time approach that properly adjusts to the process deployment requirements dynamically, i.e., the type of tasks and the available infrastructure.

Specifically, the goal of this paper is to study how to design a task offloading scheme that works intelligently and dynamically in the mist layer of satellite edge computing with consideration of the heterogeneity of applications, real-time performance requirements, movement of devices, and system-level constraints such as latency and resource availability.

Objectives & Contributions

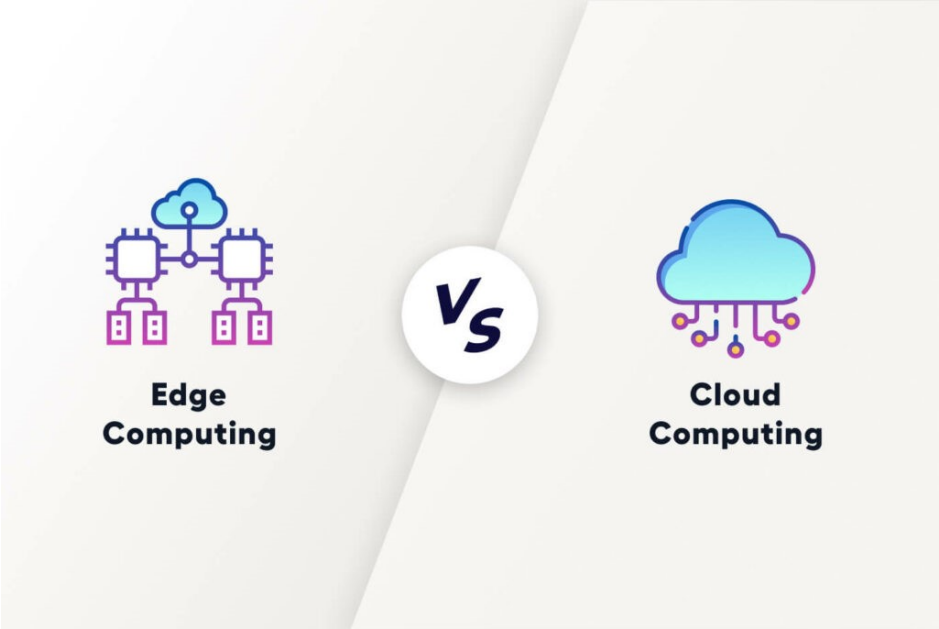
- To develop a task classification model concerning application profiles and task parameters.
- To develop a novel offloading algorithm, IsoLink, specifically designed for execution in the mist layer.
- To integrate the IsoLink algorithm within the SatEdgeSim simulator.
- To evaluate IsoLink's performance relative to existing predetermined algorithms in terms of latency, CPU load, success rate, and energy efficiency.

Thesis Outline

This thesis contains the following kind of structure:

- **Chapter I** presents a background in theory for cloud and edge computing.
- **Chapter II** provides a state-of-the-art review of satellite-edge computing, which covers task offloading.
- **Chapter III** details the design for the IsoLink algorithm, and also details IsoLink's components.
- In **Chapter IV**, the description explains the environment simulator and integrates the scheme proposed.
- **Chapter V** presents the results, which include the performance evaluation and discussion.
- The thesis is concluded and future research directions are proposed in **General Conclusion and Future Work**.

Chapter I:
Cloud and Edge Computing: Generalities



"The way to get started is to quit talking and begin doing."

– Walt Disney

1 Preface

The last few decades have transformed beyond recognition from basic computers to web-based applications to new developments such as cloud and edge computing. As styles of computing, they drive new processing and storage capabilities across increasingly connected societies. Where cloud computing suggests a more centralized style, edge computing implies a more decentralized, localized processing; yet both styles are critical to technological applications. Both offer different solutions to similar problems edge computing features faster response times and low latency, while cloud computing boasts scalability and accessibility.

Therefore, this chapter will explore cloud and edge computing, their underlying concepts, benefits and drawbacks as well as relative approaches as they exist in today's distributed world. Furthermore, we will include real-world applications of where each would be used.

2 Cloud Computing

2.1 Definition

Cloud computing is "a technology that enables convenient, on-demand network access to a shared pool of adjustable computing resources (e.g., networks, servers, storage, applications, and services) ... that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]." This essentially means that as long as users have access to the internet, they do not need to possess the underlying infrastructure as they can provision whatever options they need via the internet and utilize them as they desire (see Figure I.1).

Some essential characteristics of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. Cloud services come in three primary service models: Software as a Service (SaaS) , Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) (see Figure I.2) , as shown in Table I.1. It also employs four deployment models: private cloud, community cloud, public cloud, and hybrid cloud (see Figure I.3), as shown in Table I.2.

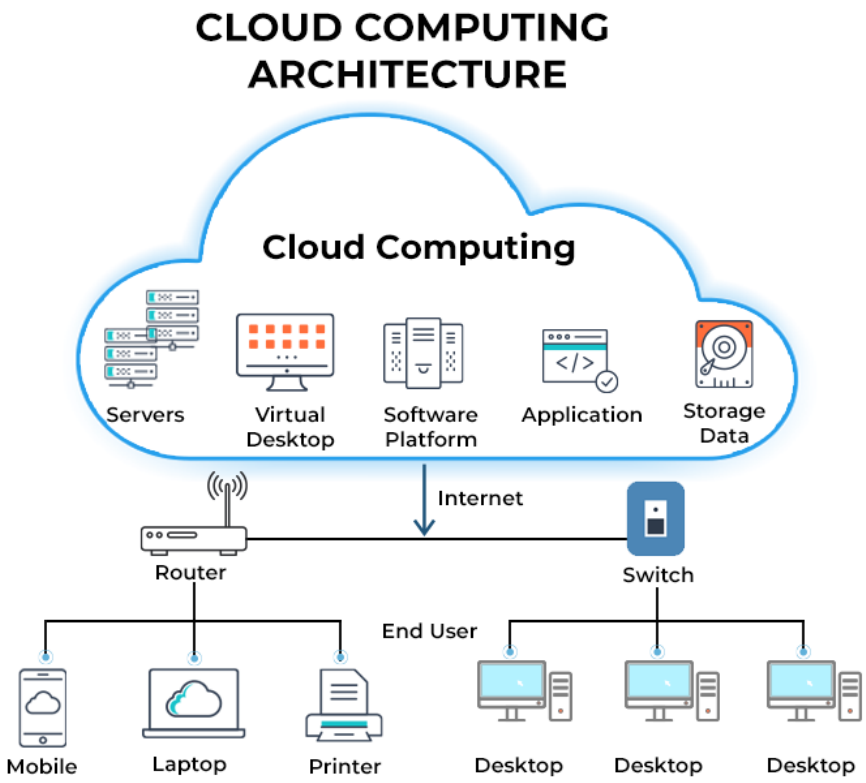


Figure I.1: Cloud Computing Architecture [2]

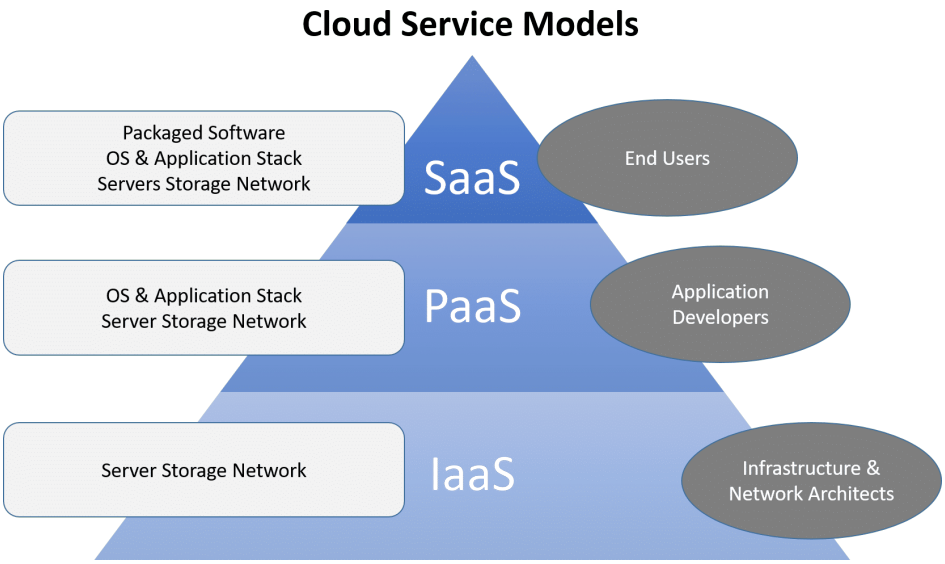


Figure I.2: Cloud Service Models [3]

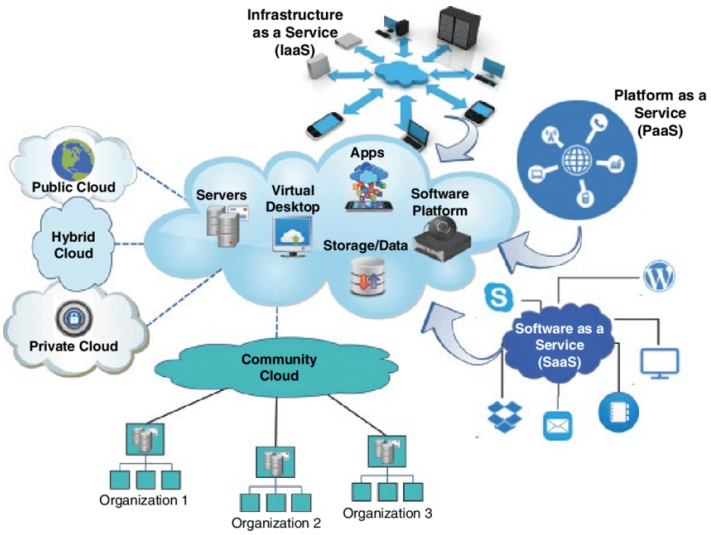


Figure I.3: Layers of cloud Architecture [4]

The three main cloud service models are [5]:

Table I.1: Cloud computing service models

IaaS (Infrastructure as a Service)	IaaS was the first proposal for a cloud service. The supplier delivers scalable hardware services in the cloud and implements a prescribed population of software frameworks through which consumers install and run any sort of software from operating systems to apps. This means that if a consumer wants to do this, they have to do more lifting on their end.
PaaS (Platform as a Service)	Platform as a Service cloud-based hardware and lower-layer software services, which provide data center infrastructure, operating systems, middleware (e.g., database systems), and a framework for hosting applications. In comparison to IaaS, PaaS has an edge when it comes to middleware and operating system pre-configuration, optimizing operation and deployment time while minimizing costs. In this case, the customer receives the cloud's scalability and efficiency while building upon the systems that the provider has already established. For any application that fits within the vendor's framework and infrastructure, PaaS ensures quicker time-to-market by guaranteeing the availability of essential components.
SaaS (Software as a Service)	Software as a Service (SaaS) forms the basis of the cloud when an application or service is offered to end-users through clients, physical devices (mobile, thin client, etc.), or the cloud. Most control of the SaaS layer is on the client side of the user, allowing for accelerated growth in industries. Examples include Microsoft Office 365, Salesforce, and Google Apps. More broadly, the SaaS model enables users to remotely access applications hosted on a service provider's cloud infrastructure via a client interface, obviating the reliance on either local installation or management.

In the cloud environment, three forms of cloud topologies are commonly used [5]:

Table I.2: Cloud computing deployment models

Private cloud	Can be described as infrastructure owned by a single business or organization, ensuring that no other tenant shares resources external to the owner's infrastructure. Resource sharing and pooling are common within the organization. Organizations adopt private clouds for security and control considerations, ensuring that only customer-managed systems can access customer-managed data. To be deemed a cloud, it should have a cloud-like quality such as virtualization and load administration. A private cloud may be hosted on-premises, off-site, or on dedicated hardware offered by a third party.
Public cloud	Is the inverse of a private cloud, providing infrastructure as needed to an open-ended variety of clients and applications. Users share and pool resources according to service-level agreements. The advantage lies in the massive scalability, which is the amount that users are willing to pay for the services to be offered, thanks to the huge number of available cloud data centers. Public cloud infrastructure is general-purpose and may be owned and operated by corporations, academic institutions, government agencies, or a combination of these. It is hosted on the service provider's premises.
Hybrid cloud	Consists of at least two clouds, either public, private, or community, and enables the simultaneous use of multiple types of clouds. This would be perfect for organizations needing specific management of sensitive data on the one hand and taking advantage of the scalability and reach of public clouds on the other. Hybrid solutions can also allow for cloud bursting, where organizations use public clouds as a backup for workloads when their private cloud infrastructure reaches its limits or is not available.In such cases, the public cloud serves as a load balancer until private cloud capacity is returned. The hybrid cloud configurations are linked through standardized or proprietary technology that enables the mobility of apps and data.
Community cloud	The infrastructure is only shared with a few selected clients from similar businesses. Infrastructure may be owned, managed, and operated by one or more community organizations, a third party, or both. It might be on- or off-site.

2.2 Main Advantages

Cloud computing has several advantages:

- Elasticity and Scalability: Cloud computing knows no bounds in terms of resources that can be provisioned and scaled to meet demand with speed, allowing businesses to manage loads efficiently.
- Cost Efficiency: Organizations only pay for what they use on a pay-as-you-go basis, saving on the costs associated with heavy investments in hardware and infrastructure.
- Optimized Resource : Compared to data centers, cloud computing gives you a higher utilization rate due to statistical multiplexing, reducing your costs.
- Flexibility: Applications can be deployed and tested quickly and without the requirement of a significant capital investment, allowing for innovation and faster product launches.
- Low Risk: As cloud computing offers flexibility, it mitigates the risk of provisioning more or fewer resources than required, eliminating wasted expense on resources that are never used or downtimes that can prove detrimental to customers.
- Accessibility: Computer resources and services are global, not local, through cloud computing, used on demand, and supported on almost all devices and applications [6] .

Such features make it a revolutionary model for businesses, leading to low operational costs and increased overall efficiency.

2.3 Major Drawbacks

While cloud computing has many advantages, it also has some disadvantages that impact reliability and the likelihood of more widespread use:

- Reliability: Major outages of the cloud service could cause significant problems, and end users have no access to the cloud infrastructure when it is down.
- Security and Privacy: Data breaches, privacy issues, and identity theft are substantial concerns with centralized cloud environments that store sensitive information.
- Reduction in Data Control: Less control over data means the user must trust the cloud provider and rely on them.

- No Specialty Standards: The inability to create and fully develop widely used specialty standards makes cloud computing less adopted and non-interoperable.
- Internet Connection Required: Cloud computing services require high-speed internet connections and stable access; they are less useful in places without adequate broadband access [7] .

2.4 Current Challenges

With the continuous advancement of cloud computing, there are many continuing problems that impact development and feasibility. Below are some of the more notable concerns that require transformative solutions to make feasibility, compliance, and performance sustainable:

- Energy Consumption: Operating massive cloud infrastructures uses up a lot of energy, raising sustainability issues.
- Data Center Locations : Users frequently have no idea where exactly their data is hosted, which leads to concerns about service reliability and downtime when cloud providers use third-party hosting.
- Regulatory Compliance: Some organizations are required by local and international laws and regulations to store and process data in specific locations, making it difficult to use cloud computing in certain industries and areas [8] .
- Advanced Security Concerns: There is growing complexity in safeguarding sensitive data, including encryption and masking, to protect against breaches.
- Vendor Lock-In: Heavy reliance on a single provider creates barriers for switching vendors.
- Cost Management: Rapidly changing pricing structures can complicate budgeting [9] .
- Latency and Performance: 5G and IoT applications require ultra-low latency to execute in real-time. Centralized cloud systems almost always fail to deliver, creating performance issues due to such high dependence. Latency must improve for effective and immediate execution [10] [11].
- Explosion in IoT and Devices:Data generation is increased and centralized systems are overloaded by the exponential growth of devices.

Such challenges create problems for the cloud computing community that need transformative solutions to ease concerns.

3 Edge Computing

3.1 The Need for Proximity Processing

Cloud computing suffers from tremendous latency because the centralized data center is too far away, generating latency issues for any real-time application, from augmented reality to driverless vehicles to some industrial IoT networks. Here is where edge computing enters to fix the issue. Edge computing is proximity processing. It takes computing and storage closer to data sources and end-users. Thus, proximity processing minimizes latency, network jitter, and even bandwidth usage while providing the performance metrics required for latency-sensitive applications. By leveraging localized nodes, proximity processing solves what traditional cloud models cannot for enhanced responsiveness and performance [12] .

3.2 Definition

Edge computing is a group of enabling technologies that perform processing on the network edge for cloud computing services upstream and IoT downstream. The 'edge' is any computing and networking resource positioned in the topology between the data source and the cloud data center. Therefore, this technology specifies where the processing occurs in near real-time, at the source (or close to it), instead of the centralized cloud resources to which every single piece of information has to travel. It experiences reduced latency and avoids unnecessary strain on the network (see Figure I.4) [13] .

3.3 Main Advantages

Edge computing offers several key advantages over centralized processing that will address the challenges explored above. These advantages include performance boosts, dependability, and efficiency in meeting tasks of a real-time nature:

- Reduced Latency: Edge computing means that data is processed on location or nearby, eliminating the delay in sending data to a centralized data center for processing, and subsequently, time-sensitive information is sent back.

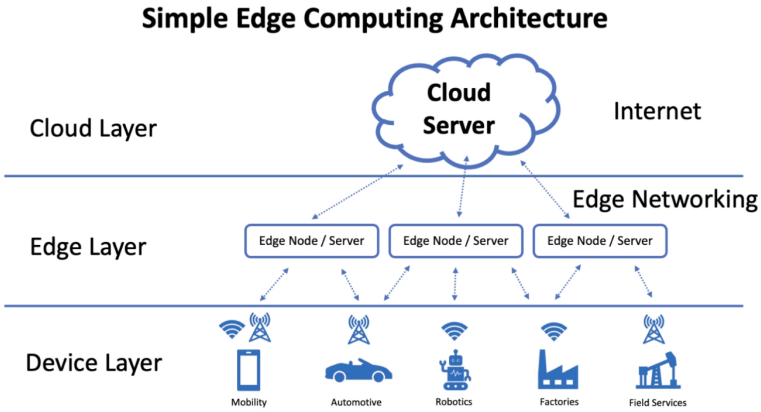


Figure I.4: Edge Computing Architecture [14]

- Better Bandwidth Consumption: Less data travels over the network to centralized data centers because less needs to go and come back when work is done at the local level.
- Increased Reliability: When a centralized data center goes down due to a networking issue, it can completely stop all data processing for a firm. However, edge devices can often function without centralized processing.
- Support for Real-Time Applications: Edge computing is conducive to real-time application support like AR/VR and IoT due to low latency and effective processing power [12] .

3.4 Major Drawbacks

Edge computing has some major drawbacks. The following issues are critical to edge computing that should be taken into consideration and solved:

- Infrastructure Complexity: The fragmented edge computing infrastructure is complicated to set up and maintain, needing vast amounts of resources in time, money, and expertise.
- Energy Consumption at Scale: While edge computing decreases the centralized power requirements of data centers, in the field, increased power requirements could be generated due to the need for multiple edge nodes to service localized areas.

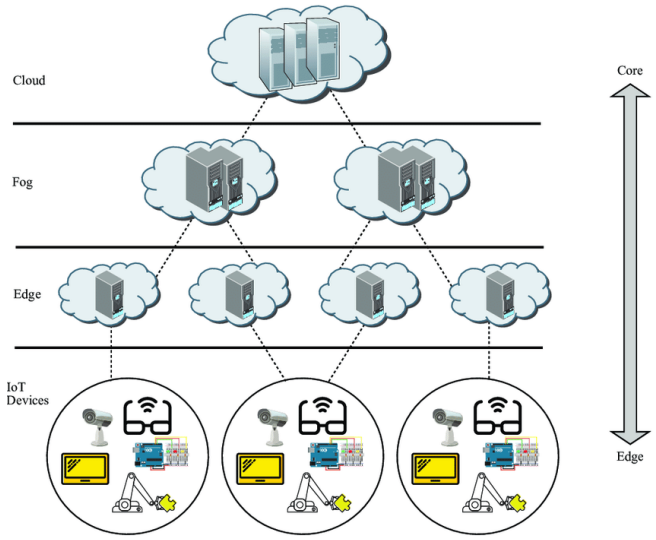


Figure I.5: Fog Computing Architecture [17]

- Security Risks: Edge computing increases security risks, as there are more nodes, and without centralization creating one attack surface, each edge node exists as an attack surface all by itself.
- Interoperability Issues: Integrating edge computing with existing centralized architectures could be complex and costly [12] .

3.5 Paradigms of Edge Computing

Edge computing is a vast and distributed system paradigm that allows for a wide array of processing, storing, and analyzing data. Yet, among these possibilities, many sub-paradigms exist in relation to functionality and application. The following are the relative sub-paradigms:

3.5.1 Fog Computing

It is a decentralized topology that allows for computing where the data resides. Fog Computing consists of a variety of node access points, IoT gateways, and routers as FCNs. By computing and caching data at the edge rather than transmitting everything to a centralized data warehouse, fog computing increases efficiency by generating resources from underused, low-latency operations due to geographical proximity to end users and improved fault tolerance when considering potential failures at the cloud level (see Figure I.5) [15] [16].

Figure I.6: MEC Architecture [18]

3.5.2 Mobile Edge Computing (MEC)

Mobile Edge Computing takes place within the mobile network base station/RAN. The major advantage of MEC is high context awareness since it operates in real-time with knowledge of the location and current network load. It is situated right next to the Radio Access Network. Thus, MEC is appropriate for latency-sensitive transactions such as video streaming and V2X because it can rapidly adjust to requests and replies (see Figure I.6) [15] .

3.5.3 Cloudlet Computing

"Data centers in a box." These are mini data centers set up close to end users for real-time processing of data. The main advantage is low latency, as they are located nearby, and virtualization of machines allows for machine resources to be allocated where they are needed. Suggested for gaming and providing cognitive assistance (see Figure I.7) [15] .

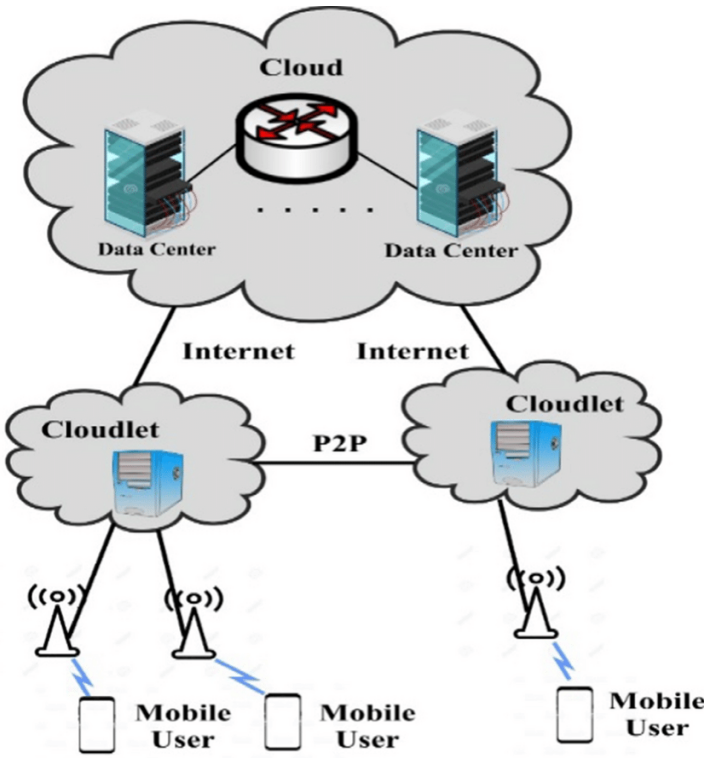


Figure I.7: Cloudlet Computing Architecture [19]

Figure I.8 represents differences between the Edge Paradigms. This comparison illustrates the main differences between these concepts in terms of the nodes used, contextual information, connectivity between nodes, and networks used.

3.5.4 IoT Edge Computing

Edge computing brings resources closer to the point of use, but IoT edge computing goes even further in the IoT device/object arena. IoT Edge Computing permits processing at the point of data creation instead of sending it to the cloud, at least not at the full processing level. Low latency, real-time processing is required for situations that need instantaneous feedback, whether from real-time monitoring, video streaming, or smart cities. Edge nodes, gateways, and mobile devices use wireless connections and on-premises computing and data caching. This minimizes requirements for the cloud in data transmission and increases security as processing occurs at the location [20] .

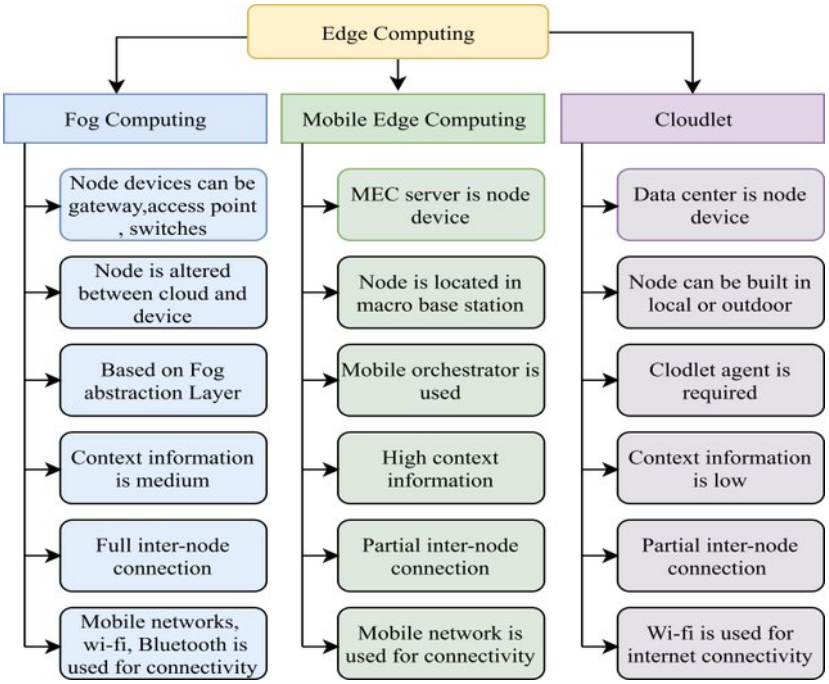


Figure I.8: Edge Paradigms Differences [18]

3.5.5 Industrial Edge Computing

Represents an important branch of IIoT technology. Data is processed instantly near industrial devices. There is less latency, which is beneficial for systems operating with instantaneous reactions like process control and real-time monitoring systems. Furthermore, less reliance on the cloud offers access for more local completion of tasks that require sensitive information, increasing security and rendering processes more seamless [21] .

4 Relationship Between Cloud and Edge Computing

Edge and cloud computing are connected because they rely on each other to create better processing of data and better-functioning systems that run and utilize this data. Cloud computing is more of a centralized, larger access to remote storage and general processing and analytical abilities. Still, it falls short in latency and bandwidth, with so many IoT devices constantly sending and receiving data and processing needs and privacy, as it needs to send and receive private data through the cloud. Therefore, edge computing is the answer. Edge computing is done where the data is at the end, meaning where it is created and where it is processed, allowing for quicker response

times, less data to be transmitted over the network, and better privacy and data security. Edge computing exists as a quasi-layer between the IoT devices and the cloud, existing more on the surface with more straightforward data and cached data existing closer to the source. However, it still needs the cloud for more extensive efforts and operations. Therefore, they rely on each other for an efficient computing environment. Such partnerships exist via hybrid means, in the center or at the edge and remotely. For example, smart cities host traffic monitoring systems that alleviate congestion in one area of a city, yet the cloud analyzes and suggests long-term adjustments to urban development. In healthcare, personal health devices provide at-home patients with quick outcomes, yet the subsequent application to the cloud offers more comprehensive diagnoses. Thus, edge and cloud computing, via hybrid means, not only enhance immediate experiences for users in the vast majority of cases but also acknowledge greater concerns and needs [13] .

5 Use Cases

5.1 Cloud Computing Applications

With web hosting and application deployment becoming easier via scalability, expansion, and contracting based on traffic, which means reduced expenses and increased efficiency and dependability, cloud computing is the future wave. It also encourages IoT and big data analytics via elastic computing power, rendering online machine learning model training often more feasible. In addition, AI software solutions require cloud services to receive the data they need to process in real-time, as the vast resources in the cloud help avoid overfitting. Furthermore, business continuity is strengthened via geo-redundant backups and redundancy features that allow almost real-time restoration of services at no additional charge; customers are charged only for what they use [6] .Moreover, data storage and backup are elastic via IaaS and Storage as a Service, which guarantees accessibility to data even in the case of local hardware failures. Improved collaboration occurs via Software as a Service that provides web-based applications and tools for communication and collaboration, regardless of the device used, which increases productivity for remote workers [1] (see Figure I.9).Figure I.10 demonstrates the use of cloud computing in healthcare applications.

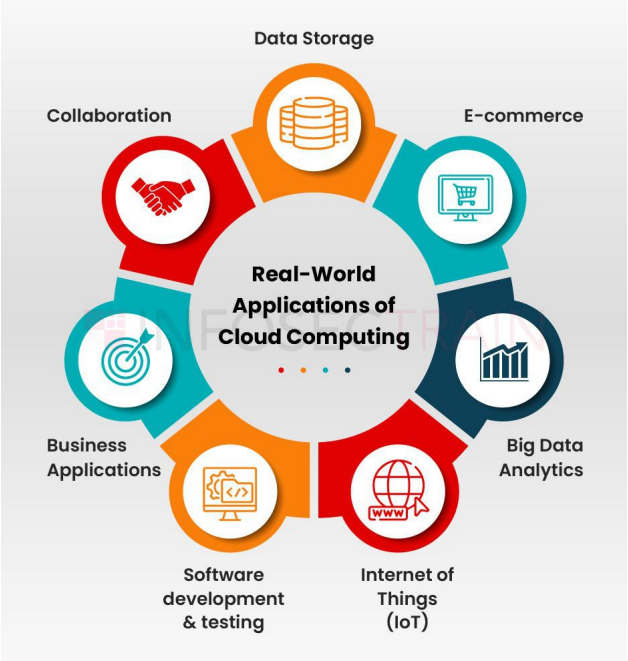


Figure I.9: Real-World Applications of Cloud Computing [22]

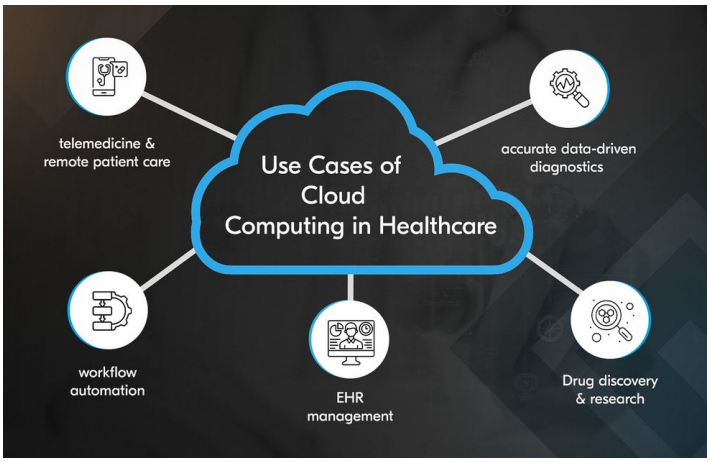


Figure I.10: Uses Cases of Cloud Computing in Healthcare [23]

5.2 Edge Computing Applications

Edge-based computing represents a total transformation of computational systems. The system operates more quickly when data no longer needs to be transmitted to a data center for processing but is handled directly on the network edge through mobile and IoT devices. This capability becomes essential for applications based on augmented reality and virtual reality because of their critical importance. Cloudlets deployed in close proximity enable faster task processing for device guidance

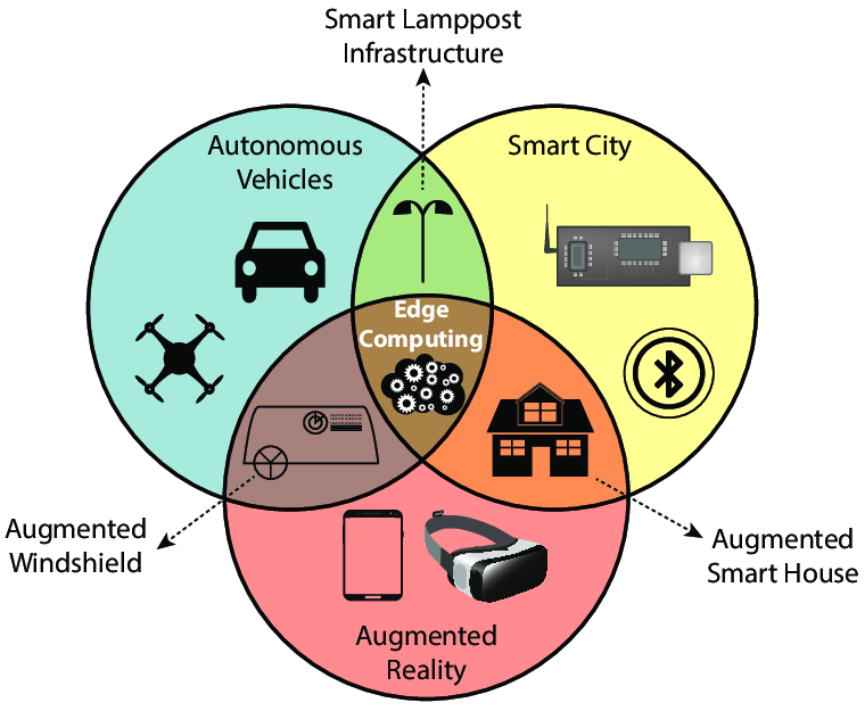


Figure I.11: Use Cases and Services Enabled by IoT Edge Computing [24]

and video analytics by performing local calculations, thus conserving bandwidth resources for other applications. Contemporary autonomous vehicles process massive sensor data that is stored in real-time to provide early detection of hazardous situations and vehicle maintenance support [13] (see Figure I.11). Smart home systems enable more responsive and private operation by letting data processing occur locally from sensors and cameras before disregarding cloud server data transfers [12] (see Figure I.12). Edge computing deployed in healthcare facilities near the local network produces faster and more reliable applications that enable quick access to patient records for better system performance.

6 Summary

In this chapter, we learned the definitions of cloud and edge computing as well as advantages and disadvantages. Cloud computing is the on-demand use of storage and computing power via the internet. Essentially, all resources are used by many people at

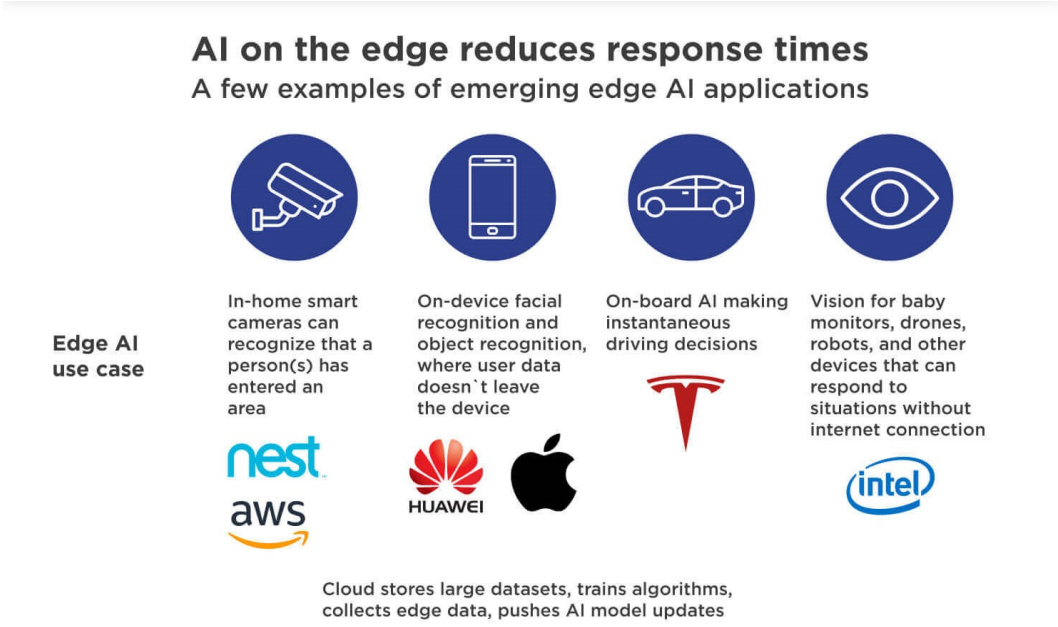
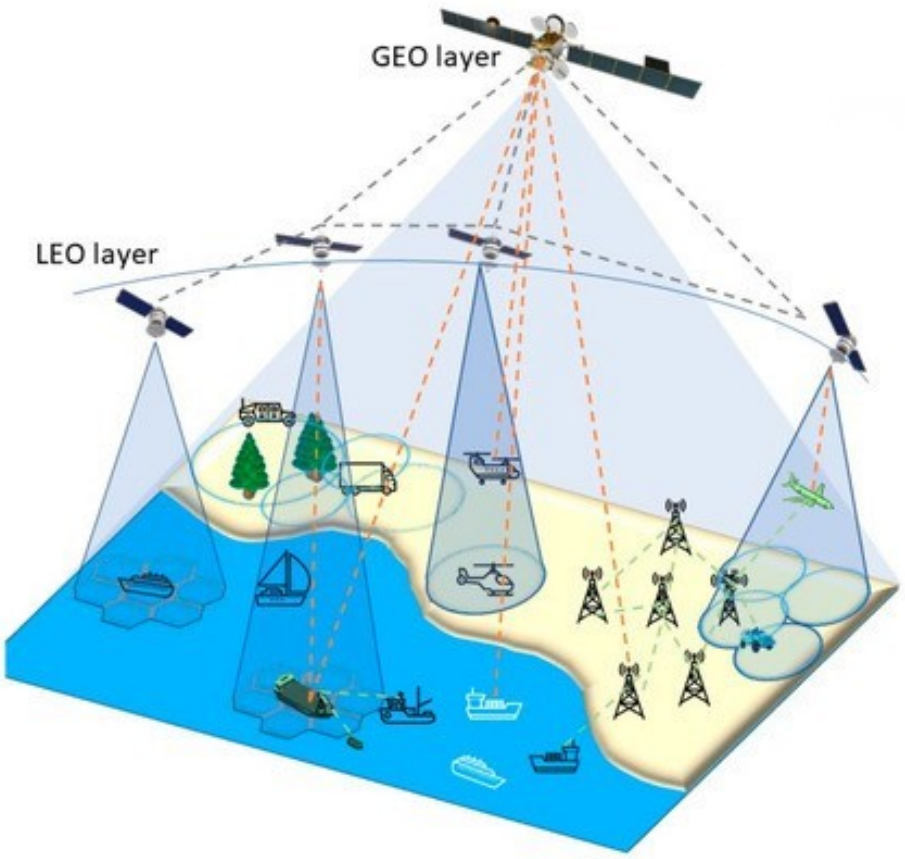


Figure I.12: AI Applications in Edge Computing [25]

once. It is a very scalable solution, cost and resource-effective due to worldwide access. Cloud computing suffers from latency, unreliable internet connections, and security concerns. Conversely, edge computing was developed to alleviate some of the challenges of cloud computing by focusing on processing data closer to the source, thereby promoting lower latency and real-time processing efforts. As such, it is particularly great for low-latency applications like self-driving cars and IoT; however, it reduces the risk of data security and improves bandwidth. At the same time, it increases the strain on infrastructure and may result in greater costs for implementation.

In the next chapter, we will build upon this idea for an edge computing solution with satellites, particularly in remote locations and disconnected environments.

Chapter II:
Satellite Edge Computing: State of The Art



"What we know is a drop, what we don't know is an ocean."

– John Dryden

1 Preface

Edge computing is a fundamental distributed system technology that improves communication bandwidth and latency by processing data locally before routing it to central servers. Among its key applications, edge computing is vital for satellite communications, which require onboard data processing without constantly depending on ground stations. Such processing ensures responsiveness in real-time, consumes less energy, improves network stability, and is critical for applications including earth observation, remote connectivity, and space-based IoT.

There have been a number of research studies on the challenges of satellite edge computing, including task distribution, resource allocation, and energy efficiency. Recent studies have proposed AI models and enhanced task offloading algorithms. In this chapter, we review the literature related to this area, covering state-of-the-art methods, challenges, and future research directions.

2 Background on Satellite Edge Computing

Edge computing has multiple applications (see Figure II.1), with one use case being satellite communications. Edge computing techniques can be used in the satellite communication field to process data in space and execute tasks in satellite systems directly rather than sending everything back to the ground stations, as seen in Figure II.2.

2.1 Evolution and Importance of Satellite Edge Computing

Satellite technology is progressing quickly. Thus, it plays a vital role in the expansion of today's communications networks. Satellite technology provides wide-area coverage and connectivity challenges within networks. Yet, it does provide access to remote locations. As edge and mist computing grow, data processing near its source minimizes transmission delay and improves effective data transmission [26].

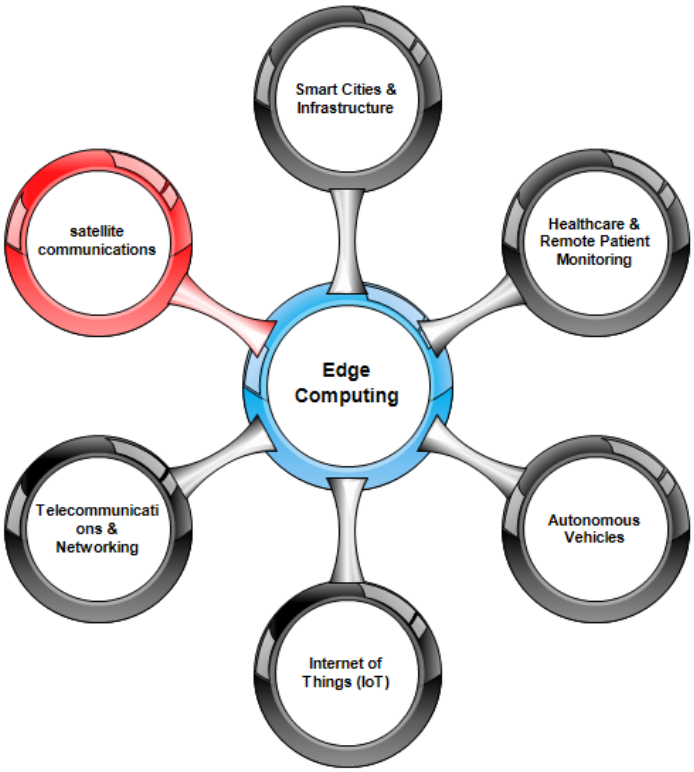


Figure II.1: Fields of Edge Computing

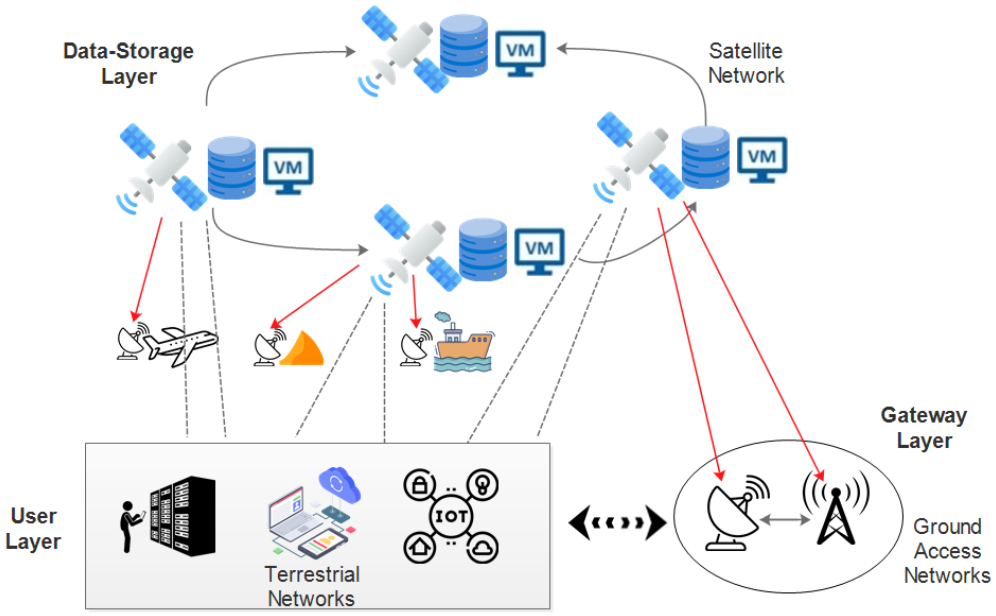


Figure II.2: Architecture of Satellite-Edge Computing Network

2.2 Challenges of Space Edge Computing

- The resources for satellite systems are restricted as they have less potential for power allocation, bandwidth allocation, and processing resources compared to terrestrial data centers.
- Satellites are in motion, causing communication lags.
- Resource allocation and scheduling needs solutions [26].

3 Related Work

To improve the performance and scalability of satellite edge computing, recent studies in the area have reported solutions:

3.1 Satellite-Based Edge Computing and Resource Allocation

Much research has been conducted on LEO satellites relative to satellite-based edge computing. For example, Wang et al. [27] solved the resource allocation challenge of an ECS by using a three-layer network topology based on a SDN modeling framework. The study applied the AKG and BFST simultaneously for effective resource allocation and ISL construction. Simulation results confirmed that the dynamic resource scheduling method was both practical and effective.

3.2 Integration of Satellite and Terrestrial Networks in 6G Architectures

Zhang et al. [28] proposed a 6G wireless architecture to integrate satellite and terrestrial networks. They proposed a new paradigm of double-edge intelligent integrated satellite and terrestrial networks, which is based on communication, storage, and computation synergy. The DILIGENT infrastructure utilizes multi-access edge computing and AI for adaptable network operation. It also investigated task offloading and content caching strategies, showing performance superiority over existing integrated networks.

3.3 Edge Computing for IoT in Aerospace and Remote Areas

Wang et al. [29] detail the rise of IoT in the aerospace industry, where edge processing able to mitigate complications caused by network distances and remote IoT installations. The solution is that existing satellites should be retrofitted as edge computing nodes in space capable of dynamic software loading, resource sharing, and

functioning in concert with cloud computing edges. Simulations reveal that time and energy savings are impressive and less than those of a traditional satellite constellation with time savings increasing based on the number of satellites used and task-offloading approaches.

3.4 Collaborative Satellite Networks for IoT Computing

Gao et al. [30] examined the use of satellite networks to complement terrestrial networks, aiming to provide connectivity to IoT users in remote areas. They suggested splitting the IoT computing task into parts and then using a network of satellites to process in collaboration. They explored how to combine NFV technology and satellite edge computing and finally introduced a VNF placement strategy that may become a potential solution through a game-based approach. It is found that the proposed decentralized resource allocation algorithm PGRA is effective in minimizing deployment costs and maximizing computing services.

3.5 Onboard Satellite Data Processing Using Edge Computing

Bui et al. The work presented in [31] is centered on data processing solutions onboard the satellites using edge computing. This study employs deep learning techniques to provide benefits for low-light image enhancement, which can enhance accuracy for early detection and more accurate tracking. To overcome the computational burden, an edge-computing-enabled inference model is proposed with an architecture composed of an encoder-decoder and illumination mapping optimized. On the other side, this model, running in Arm Cortex-M3 microcontrollers onboard satellite payloads, showed significantly faster running times compared with the conventional convolution models with no compromise on the image processing quality.

3.6 Task Assignment Optimization in Satellite-Enabled MEC

Computation offloading is one of the most important areas in Satellite Edge Computing and is designed for workload balance between terrestrial devices and satellite-based computing nodes. Most of the previous studies are based on MEC systems; however, the dynamic characteristics of the satellite network environment require particular solutions dedicated to these networks. Wang et al. [32] proposed a game-theoretic model for maximizing computation offloading in SEC and addressing the intermittent communication between satellite and terrestrial devices. Their study described the offloading process as a non-cooperative game in which each device makes

independent decisions about its offloading strategy to minimize latency and energy consumption. An almost cyclic strategy was theoretically analyzed based on the nash equilibrium of the system to obtain the optimal offloading strategies using iteration.

3.7 Dense Satellite-Terrestrial Integrated Mobile-Edge Computing Networks

Ding et al. [33] propose that the architecture of a dense SATIMECN satisfies the computing requirements for the upcoming generation of networks. Based on this, the study constructs an energy consumption minimization problem, considering user-satellite association, task allocation, and computation resource allocation. Using Lyapunov optimization theory and the idea of decomposing the optimization problem, the proposed approach can obtain a tradeoff between energy consumption and queue length, which works to optimize computation offloading efficiency in dense satellite networks.

3.8 Satellite Mobile Edge Computing for Earth Observation Data

Leyva-Mayorga et al. [34] discuss the issues associated with the management of high volumes of information derived from earth observation satellite imagery transmitted with LEO satellites. Specifically, this study devises a SMEC framework in which the strategies of image distribution and compression parameters are optimized to minimize energy consumption while ensuring real-time processing capabilities. Results show major advancements in both capacity and energy efficiency, especially in various forms of real-time disaster recognition and response.

3.9 Resource Optimization in Mist Computing

Babaghayou et al. [35] introduce a proximity-oriented low-layer orchestration algorithm that aims to enhance resource allocation in mist computing environments. Mist computing, a further evolution of edge computing, aims to bring computing resources even closer to the location where the mission is prevalent, often at the network's extreme edge. APOLLO solves resource constraints, latency, and energy conservation problems through proximity-based resource orchestration. This paper provides a study on how edge applications can improve their resource utilization and system performance, and serves as a valuable reference for the field of distributed and edge computing.

3.10 Task Offloading in Satellite-Enabled Mist Computing

Babaghayou et al. [36] achieve a proximity-aware and load-balancing direction-sensitive task offloading algorithm to improve the performance of satellite-related mist computing environments. Satellites and mist computing nodes are connected in a distributed computing system where the nodes have computing resources distributed throughout the system. It enhances resource utilization, reduces latency, and improves the performance of the system overall by making good use of proximity-based task allocation and load-balancing techniques. Additional simulations demonstrate the performance of their approach, indicating that it is adequate for real-time applications in satellite-enabled mist computing systems.

Table II.1: Comparison of Previous Research in Satellite Edge Computing

Research Area	**Objectives**	**Techniques Used**	**Benefits**
Satellite Edge Computing and Resource Allocation [27]	Improve resource allocation in satellite edge computing	SDN, AKG and BFST algorithms	Reduce latency, improve communication efficiency
Integration of Satellite and Terrestrial Networks in 6G [28]	Integrate satellite and terrestrial networks in a 6G environment	MEC, AI, Task Offloading, Content Caching	Improved network management, reduced latency
Edge Computing for IoT in Space and Remote Areas [29]	Enable IoT in remote environments through edge computing	Dynamic software loading, resource sharing, cloud coordination	Reduced energy consumption, improved response time
Cooperative Satellite Networks for IoT [30]	Enhance cooperation among satellites to support IoT	NFV, Game Theory, Task Division among Satellites	Improved performance in remote areas

Onboard Data Processing Using Edge Computing [31]	Accelerate image processing in satellites	Deep Learning, Encoder-Decoder Architecture, Illumination Optimization	Improved space image quality, reduced power consumption
Task Allocation Optimization in Mobile Edge Computing via Satellites [32]	Optimize task offloading to reduce power consumption	Game Theory, Nash Equilibrium, Iterative Algorithm	Reduced latency, improved power efficiency
Mobile Edge Computing in Dense Satellite-Terrestrial Networks [33]	Balance energy consumption and system performance	Lyapunov Optimization, Task Allocation, User-Satellite Association	Reduced power consumption, improved performance
Satellite Mobile Edge Computing for Earth Observation Data [34]	Optimize distribution of satellite-broadcast Earth observation images	Data compression, image distribution optimization	Reduced energy consumption, enhanced disaster response applications
Resource Optimization in Mist Computing [35]	Improve resource allocation in mist computing environments	APOLLO Algorithm, Low-Layer Orchestration	Reduced latency, improved resource allocation efficiency
Task Offloading in Satellite-Assisted Mist Computing [36]	Achieve balanced task distribution between mist and satellites	Distance-Based Task Allocation, Performance Enhancement	Reduced latency, improved system performance

We materialize a systematic taxonomy in Figure II.3, which classifies the previous research concerning satellite edge computing into five major areas: task offloading, energy efficiency, latency, resource management, and applications:

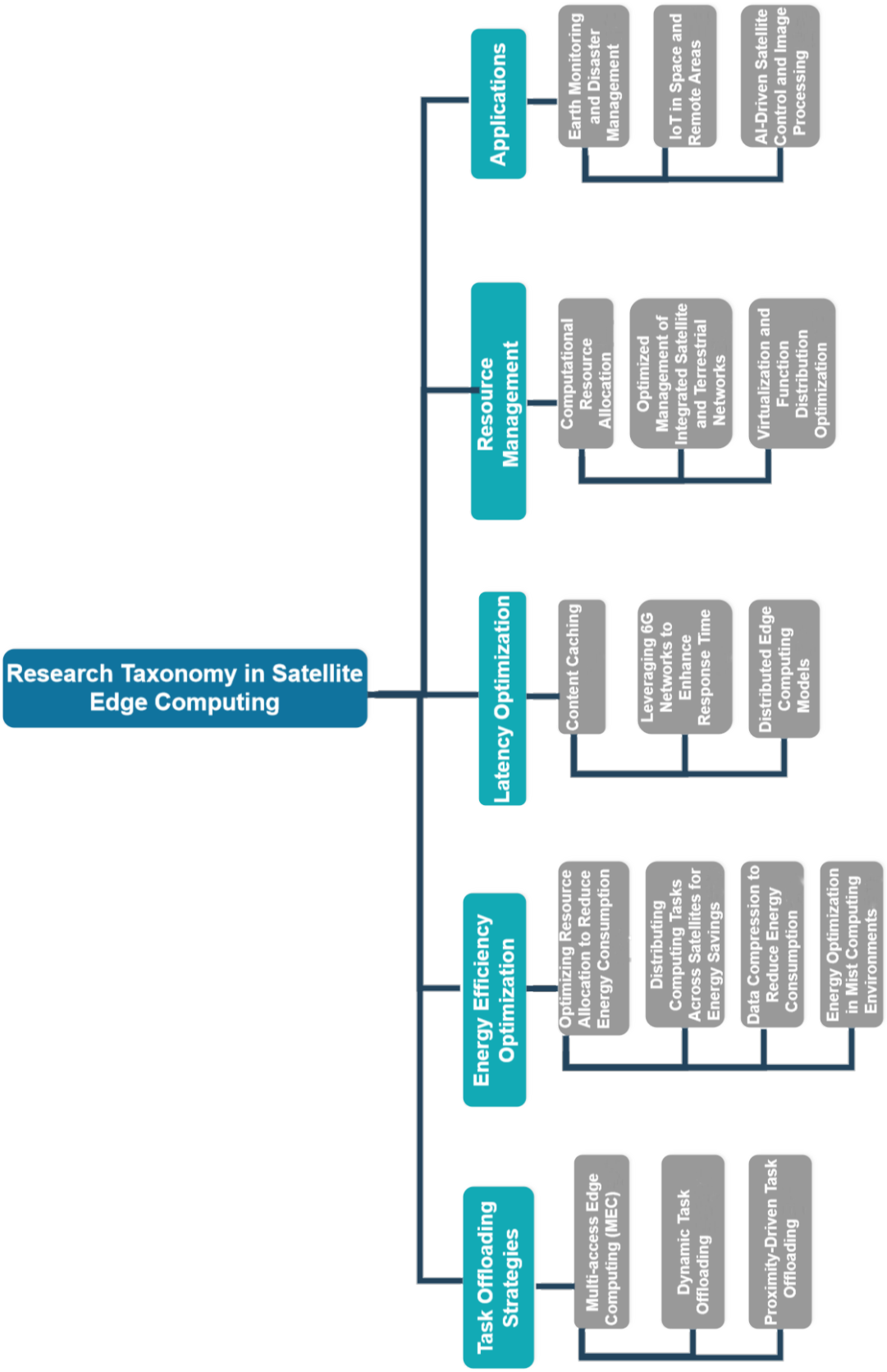


Figure II.3: Taxonomy of Research in Satellite Edge Computing

4 Summary

Satellite networks have seen a turning point by integrating edge computing with them. We conducted a literature review and presented key research contributions focusing on task allocation, energy efficiency, and network optimization of satellite-enabled edge computing. The recent developments have presented AI-based models, game-theoretic methods, and SDN techniques that improve resource allocation, latency reduction, and scalability. However, challenges remain for further exploration.

In the next chapter, we will explain our approach that enhances and focuses mainly on end-to-end delay while still keeping the other metrics like energy, processing time, and functioning at an acceptable level.

Chapter III:
Our Proposed Task Offloading Approach

"In the middle of every difficulty lies opportunity."

– Ralph Waldo Emerson

1 Preface

In light of advances in distributed computing and the growing need for latency-critical applications, mist computing is essential for offloading ground IoT devices' workload on adjacent satellites. This chapter presents the proposed offloading algorithm, **IsoLink**, specially designed for managing mist-level processing in satellite-based edge systems.

2 System Model and Assumptions

2.1 Entities Involved

The architecture proposed combines three satellite tiers with ground IoT devices (see Figure III.1) :

- **Cloud Satellites:** High-capacity satellites with considerable communication delays.
- **Edge Satellites:** Satellites with intermediate delay and resources.
- **Mist Satellites:** LEO satellites with the highest ground proximity, providing the lowest latency and best suited for real-time applications.

Ground IoT devices are responsible for creating applications-driven tasks in areas like augmented reality, e-health, smart farming, IoT sensors, smart cities, machine learning, autonomous vehicles, and online video streaming.

2.2 Assumptions

For simulating the environment and developing the IsoLink algorithm, the following assumptions are made:

- We did a pre-processing to adapt the number of satellites to our isolated environment.
- Tasks are classified into real-time, latency-tolerant, or heavy, based on size, deadline, and complexity.
- The task offloading decisions are limited to the mist layer to minimize communication latency.

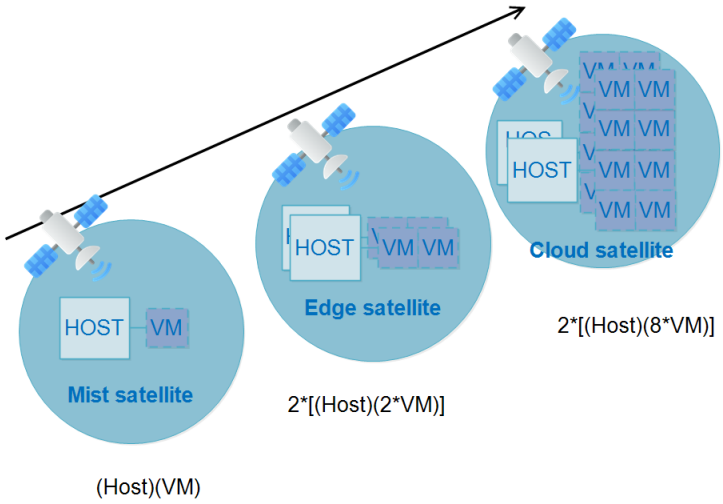


Figure III.1: The assumed types of satellites and their resources.

- Application profiles (e.g., required cores, deadlines, size of the data) are specified in an XML document and employed in task categorization.
- Distance, VM load, and MIPS are taken into account when determining the offloading cost.

3 IsoLink - the Novel Task Offloading Scheme

IsoLink is an application-tailored and latency-conscious offload mechanism. It uses task categorization and normalization of metrics for informed VM selection that allows for efficient processing within the mist layer. It emphasizes responsiveness by assembling task needs with local resources with minimal delay, striking a balance in computational load distribution on satellite-based infrastructure.

The IsoLink algorithm achieves efficient and knowledgeable offloading in the mist layer through core steps:

1. **Task Profiling:** Every task is assigned the most appropriate predefined application.
2. **Classification:** The task belongs to one of three categories.
3. **Weighting by Task Type:** Dynamic weights are assigned according to task type.

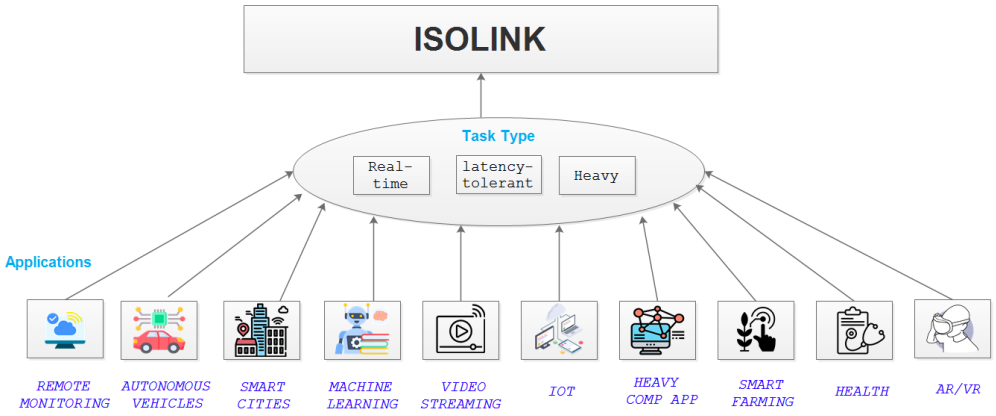


Figure III.2: Task Classification and Application Model Matching in IsoLink

4. **Metric Normalization:** Execution delays and communication delays are normalized for candidate VMs.
5. **Score Evaluation:** A weighted score is calculated for every VM.
6. **VM Selection:** The VM with the lowest cost is selected.

Figure III.2 depicts the task processing's starting stage in the IsoLink offloading algorithm. It begins with Task Profiling, in which each task is compared with the pre-defined application profiles in an XML file in order to identify the task that best matches.

Then, the task goes through classification, in which it is classified into one of three broad types: real-time tasks, latency-tolerant tasks, and heavy tasks.

The categorization is dependent on container size, the number of CPU cores needed, task duration, and delay sensitivity. This sequential process is the basis for intelligent offloading decisions, with each task handled according to its nature and need for execution.

4 Pseudo-Algorithm Design

This section provides the formal pseudocode of the IsoLink algorithm, explaining task categorization, delay, and execution calculation, and how the best virtual machine

is chosen for offloading.

Algorithm 1: IsoLink

Input: Task `task`, VM List `vmList`, Application List `applications`, Architecture `architecture[]`
Output: Index of selected VM or -1 if no valid VM is found

```
function IsoLink (task, vmList, applications, architecture);
if task == null then
    return null
end
Load application profiles from XML into applications;
matchedApp ← classifyAndMatchApplication(task, applications);
taskType ← classifyTaskType(task);
task.type ← taskType;
taskTypeCounters[taskType] ← taskTypeCounters[taskType] +1;
taskApplicationCounters[matchedApp] ← taskApplicationCounters[matchedApp] +1;
Initialize disdelay[], exedelay[], vmnum[];
for i ← 0 to vmList.size() - 1 do
    vm ← vmList[i];
    disdelay[i] ← distance(task.device, vm.device) / networkSpeed;
    exedelay[i] ← task.length / vm.MIPS;
    vmnum[i] ← number of tasks assigned to vm;
end
Normalize disdelay[] and exedelay[] to disdelay_stand[] and exedelay_stand[];
if taskType == "real-time" then
    w_delay ← 0.7, w_exec ← 0.3;
end
else if taskType == "latency-tolerant" then
    w_delay ← 0.4, w_exec ← 0.6;
end
else
    w_delay ← 0.3, w_exec ← 0.7;
end
bestVM ← −1, minScore ← ∞;
for i ← 0 to vmList.size() - 1 do
    if vmList[i] is valid for offloading then
        score ← w_delay· disdelay_stand[i] +w_exec· exedelay_stand[i];
        if score < minScore then
            minScore ← score;
            bestVM ← i;
        end
    end
end
return bestVM;
end function
```

1. **classifyAndMatchApplication (Task task, List applications):**
   - **Purpose:** This function finds which application profile matches the incoming task the best.
   - **How it works:** It compares all parameters from maximum delay, container size, required cores, and length of task. Each of these attributes adds up to a comparison score, and the application with the lowest score is determined to be the best match.
2. **classifyTaskBasedOnProperties (double maxDelay, double containerSize, int requiredCore, double taskLength):**
   - **Purpose:** Classify the task into one of three categories: real-time, latency tolerant, and heavy.
   - **How it works:** This relies on required thresholds. For example, if $maxdelay \leq 10$ seconds and container size and required cores are all less than 2, then classify it as a "real-time" task; otherwise, if not, it falls into the other classifications.
3. **readApplicationsFromXML () :**
   - **Purpose:** To read the application configuration file (XML) and compile a list of application profiles.
   - **How it works:** It parses each tag reading max_delay, container_size, task_length, and required_core information to create application objects from those values stored in a list.
4. **standardization (List metricList):**
   - **Purpose:** Standardize metric values, such as execution time or delay.
   - **How it works:** Min-max formula is used:

$$\text{standardized\_value} = \frac{\text{value} - \min}{\max - \min}$$

   So that all metrics can be comparable as a value between 0 and 1.

5. **offloadingIsPossible (Task task, Vm vm, String[] architecture):**
    - **Purpose:** To determine whether it is possible to offload this task to this VM given these conditions.
    - **How it works:** This typically requires checking whether the VM has enough resources (CPU, Memory), whether the network path is valid, and for now, if the system architecture allows for such offloading.

5 Summary

In this chapter, we have proposed the IsoLink algorithm, a lightweight and efficient offload strategy for mist-level execution. It uses task categorization, application matching, and dynamic cost evaluation in order to select the optimal executing node.

In the next chapter, we discuss the simulated environment, explaining its properties and tools, and how we integrated IsoLink into the framework. Then, we will evaluate the performance of IsoLink through simulation in SatEdgeSim and compare it with simple offloading algorithms by means of several performance criteria, including the delay of task execution, task success rate, and resource utilization.

Chapter IV: Simulation Environment for Satellite Edge Computing

"The purpose of simulation is insight, not numbers."

– Richard W. Hamming

1 Preface

Exponential space technology developments led to the creation of advanced communication systems that operate with higher efficiency and reliability. The implementation of edge computing throughout space-based infrastructure requires specialized simulation tools, as its adoption has grown significantly. CloudSim and EdgeCloudSim function as robust simulators for cloud and terrestrial edge computing, yet they lack the specialized capabilities needed to exactly represent satellite edge computing dynamics.

The development of SatEdgeSim represents a complete toolkit that helps model and evaluate satellite edge computing infrastructure. The SatEdgeSim framework enables research studies about task offloading strategies and network efficiency, as well as energy usage because it includes features for orbital mobility in conjunction with dynamic network structures and resource limitations. This chapter investigates present-day simulator limitations while describing the SatEdgeSim framework and demonstrating its ability to simulate real-world satellite computing systems.

2 The specialized simulator for satellite edge computing

The developers created SatEdgeSim to extend PureEdgeSim functions (see Figure IV.1) by adding enhanced simulation capabilities for satellite edge computing situations, by providing a simulation environment that considers orbital motion dynamics, task offloading strategies, and energy consumption under the limited resources of satellites. The simulator enables researchers to experiment with new models to improve their technologies and designs, making future space networks more efficient [26] .

3 SatEdgeSim's Architecture and Foundations

3.1 SatEdgeSim's Fundamental Elements

Essential elements of the SatEdgeSim system allow for the simulation of satellite edge computing environments [26] (see Figure IV.2) :

3.1.1 Simulation Manager

- Manages all aspects of simulation and event processing.

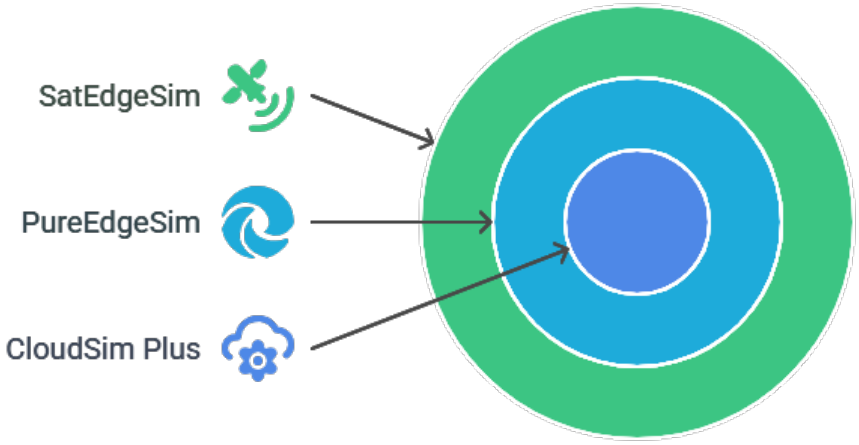


Figure IV.1: Simulation Framework Hierarchy

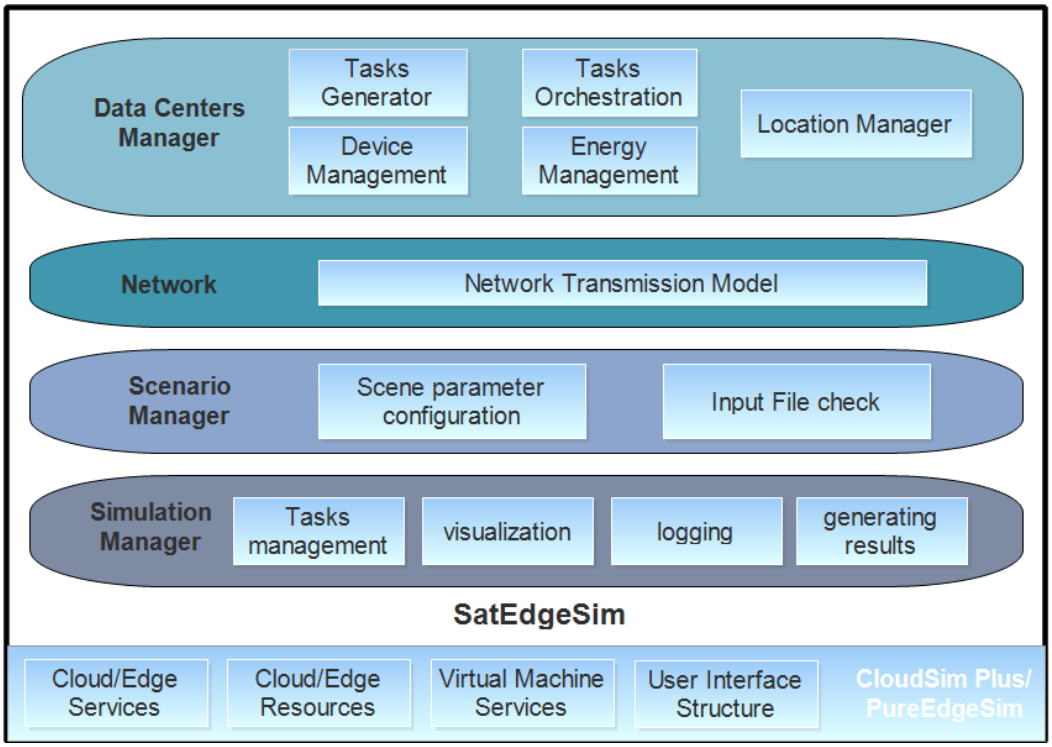


Figure IV.2: SatEdgeSim block diagram

- Manages data processing, task flow, and performance evaluation.
- The module generates reports and analyzes simulation results, including task delay data, success rates, and energy consumption measurements.

3.1.2 Data Centers Manager

- The manager works in the cloud, edge, and mist segments across various computing resource domains.
- By using their deployed task allocation parameters, system managers can supply resources for satellites and edge nodes.
- The integrated model provides a method for evaluating the energy efficiency of the entire system.

3.1.3 Generator of Tasks

- The system generates a large number of processing assignments that operate in the satellite computing environment.
- The Tasks Generator component replicates real-world task arrival patterns using the poisson distribution.
- The task generator offers dynamic connectivity between task type data, device information, and arrival time.

3.1.4 The Location Manager

- Allows for realistic network dynamics simulation by tracking satellite movement.
- Makes use of STK data to precisely position satellites in 3D.
- Users can customize their own orbits to suit the needs of specific applications.

3.1.5 Network Module

- This module simulates ground-to-space and intersatellite communications.
- Takes mobility effects, bandwidth limitations, and signal propagation delay into account.
- Simulates how satellite movement causes dynamic changes in network topology.

3.1.6 Tasks Orchestration

- Using the current operating conditions as a foundation, the framework allows for automated task distribution among satellites.

Figure IV.3: Satellite edge computing network architecture

- When choosing deployment solutions, the decision framework considers both satellite positions, processing power, energy consumption, and delay times.
- Depending on their unique needs, users can use the system to implement personalized scheduling algorithms for task management.

3.1.7 Scenario Manager

- Sets up the simulation by entering the Earth's radius and the satellites' orbits and physical characteristics.
- Oversees the various simulation scenarios' resource allocation plans.

3.2 Cloud, Edge, and Mist Node Representations

SatEdgeSim's resource management system makes use of cloud-edge-mist architecture in its monitoring infrastructure [26], as illustrated in Figure IV.3. The three representation scheme of operating layers is as follows:

3.2.1 Nodes in the cloud

- Stand in for cloud servers with a large capacity.
- The satellites are in high-altitude orbits, ranging from 3500 to 3600 kilometers to facilitate functions related to centralized processing.
- Managing complex data processing tasks falls within the cloud node's scope that cannot be carried out at either the edge or the mist layers.

3.2.2 Nodes at the Edge

- Data centers must be positioned in medium-altitude orbits, which reach a height of 2150 kilometers.
- The system uses a mediocre level of computing power to process data at the source, which speeds up reaction time.
- Real-time data processing occurs through intermediaries that work between mist nodes.

3.2.3 Nodes of Mist

- Low-power satellites are located in the orbit between 550 and 1325 km units of computation.
- Latency-sensitive tasks and local data filtering operations are carried out with this layer.
- Because of their mobility, these units are appropriate for applications that require real-time operations with dynamic task allocation.

4 Simulation Parameters in SatEdgeSim

SatEdgeSim supports several simulation and scenario parameters that decide how the model runs and the various satellite edge computing environments that are assessed. These parameters fall under three categories: environment setup parameters, network settings, and resource consumption metrics [26].

4.1 Simulation Setup and Environment Parameters

1. Simulation Time
2. Node and Station Count
3. CPU Utilization and Energy Update Time

4.2 Parameters of the Network and Communication

1. Bandwidth and Frequency Range
2. Interval of Satellite Position Update

4.3 Consumption Parameters for Energy and Resources

1. Energy Usage Model by Type of Node
2. Model of Resource Management

5 Scenario Parameters in SatEdgeSim

SatEdgeSim contains a variety of task deployment methods across different nodes (cloud, edge, mist) to improve performance, latency, and energy consumption [26]. The key strategies are:

1. **Round_Robin Scheduling (ROUND_ROBIN)**
   Without taking into account which nodes are busier, Round Robin assigns tasks to each node in a sequential manner, one after the other. The next node on the list receives each new task. It is quick to set up, but it ignores the nodes' capabilities, which could lead to bottlenecks at some nodes and underutilization at others.
2. **Trade_Off Scheduling (TRADE_OFF)**
   By giving each node a score based on its estimated latency and energy consumption, the scheduling seeks to minimize both delay and energy consumption. The node that has the best balance is chosen. Task allocation may be slightly delayed by the computational overhead introduced, even though efficiency is increased.
3. **Traditional Polling Scheduling (TRADI_POLLING)**
   Before assigning tasks, TRADI_POLLING polls every node that is available. As opposed to Round_Robin, which assigns tasks blindly, it chooses the best node based on availability and resource status. This prevents overloaded nodes, increasing efficiency. Polling adds additional delays, though, particularly in large networks.
4. **Weighted Greedy Evaluation (WEIGHT_GREEDY)**
   A weighted formula that selects the best node by considering latency, energy, load, and processing power. The node that scores the lowest overall is chosen. While it saves energy and delays, it requires complex calculations and exact weight settings.

6 Performance Evaluation

We evaluate performance for SatEdgeSim using key metrics that capture the effectiveness of the satellite edge computing environment. Latency, energy consumption, and task success rate are examined in the simulation for improvement in task deployment schemes and enhancement of network stability [26].

1. **Task End-to-End Delay**
   Timed from sending a task from the source node to receiving a message from the destination node.
2. **Energy Consumption Rate**
   Evaluates the energy used by each node (satellite, edge center, and mist device) when performing tasks.
3. **Task Success Rate**
   The percentage of tasks that were executed successfully and not lost either because of wait time or network loss from or to the IoT Gateway.
4. **Task Failure Rate**
   The percentage of tasks that failed due to delay or satellite movement.
5. **CPU Usage**
   how much of the satellite CPU resources are used for processing.
6. **Network Usage**
   How much data is sent and how much bandwidth is used when nodes are communicating.

7 Simulation Framework Evolution and Integration of the Proposed Scheme

Figure IV.4 outlines the hierarchical composition of the simulation frameworks used within this work. At the center is CloudSim Plus, an extensible and modular cloud-computing simulator recognized for its well-designed object-oriented architecture and ability to support multithreaded runs. This is built upon by PureEdgeSim, which extends features specifically for edge and fog environments, such as mobility simulation, energy usage estimation, and simple network behavior simulation.

SatEdgeSim builds on this tradition by implementing an integrated

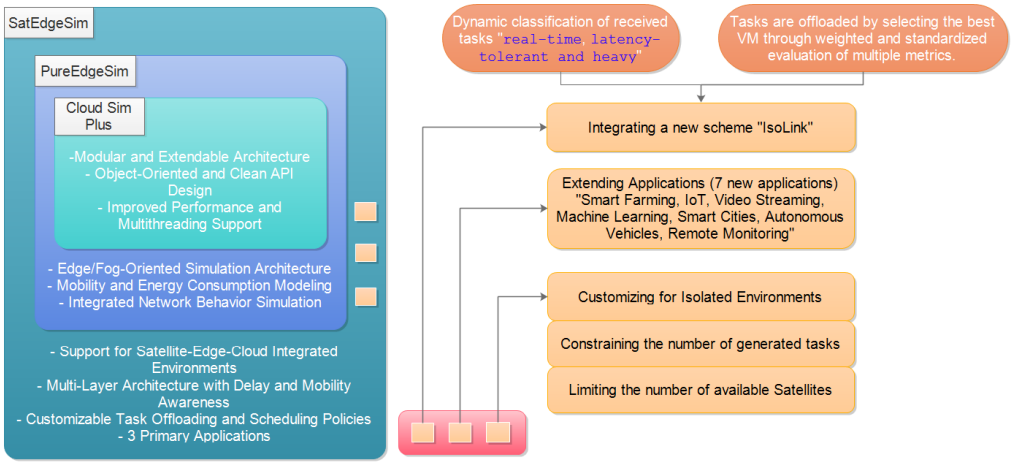


Figure IV.4: Simulation Framework and Integrated Extensions in SatEdgeSim

satellite-edge-cloud simulation platform. It features an architecture consisting of multiple layers, has intrinsic delay and node mobility awareness, supports task offloading and scheduling policy customization, and is hence apt for assessing large-scale and distributed geo-computing scenarios.

In this paper, SatEdgeSim is used as a basis for incorporating the proposed scheme, which involves dynamic task classification (into real-time, latency-tolerant, and heavy tasks) and smart task offloading, which considers virtual machines under criteria like delay, execution time, and system load. This custom logic "IsoLink" has been added to the simulator, and its scope has been extended to cover applications like smart farming, IoT services, machine learning, smart cities, and autonomous systems, allowing for the simulation of task deployment under realistic conditions within heterogeneous and resource-constrained environments (as shown in Figure IV.5 and Figure IV.6).

8 Summary

The combination of edge computing and satellite networks poses novel challenges demanding innovative and domain-specific simulation solutions. This chapter has emphasized the necessity for tailored tools and shown how SatEdgeSim overcomes the weaknesses of traditional simulators. With support for real-world modeling of satellite mobility, resource allocation, and complex task scheduling, SatEdgeSim is an effective tool for evaluating edge-satellite computing environments. The incorporation of the presented task offloading algorithm into SatEdgeSim was also explained. Now that

Figure IV.5: Overview of the Main Extensions Introduced to the SatEdgeSim Framework

the simulation framework is fully set and the custom extensions activated, it is the next logical step to evaluate the performance of the proposed solution under different scenarios and workloads.

Ultimately, the reader will find in chapter 5 an elaborate assessment of the simulation results: execution delay, energy efficiency, and successful task completion percentage relative to like algorithms as a means to assess the benefits and shortcomings of the solution offered.

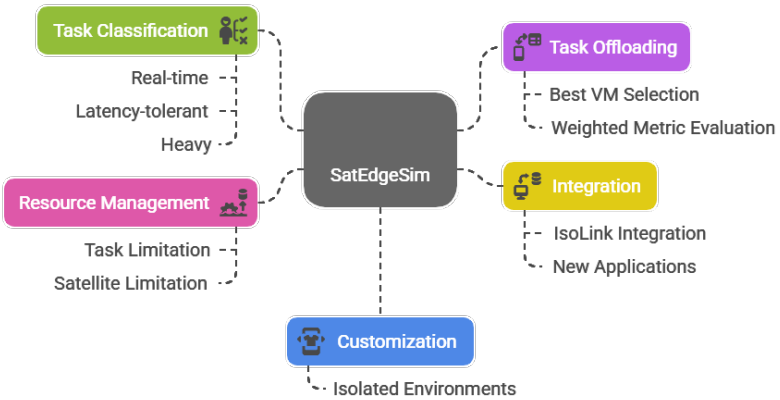


Figure IV.6: IsoLink-Aware Task Management and Resource Integration within SatEdgeSim

Chapter V:
Results and Discussion

"Without data, you're just another person with an opinion."

– W. Edwards Deming

1 Preface

In scientific research, practical experimentation in testing a proposed algorithm is as vital as designing it theoretically. This chapter documents the findings resulting from testing the **IsoLink** offloading algorithm in the SatEdgeSim simulation framework. The efficiency of IsoLink under different scenarios is measured and discussed and compared to offloading strategies to demonstrate its effectiveness in mist-level satellite-edge scenarios. We examine, through in-depth analysis of graphs, tables, and metrics, the execution delay, task success rate, and resource usage, to see how IsoLink performs in dynamic task scenarios. The objective is to uncover developing trends and show the potential of the proposed algorithm in facilitating intelligent offloading decisions.

This chapter aims not only to justify the proposed approach but also to illuminate the real-world implications of intelligent offloading in edge systems assisted by satellites.

2 Simulation Parameters

In order to verify the performance of the introduced IsoLink task offloading algorithm, a set of experiments was performed using the SatEdgeSim framework. The experiments set out to test the behavior of the algorithm in a variety of satellite-enabled edge computing scenarios. The main simulation parameters and system configurations used in all experiments are reported in Table V.1.

Table V.1: Simulation Parameters

Parameter	**Value**
Number of Mist Satellites	500
Number of Edge Datacenter Satellites	24
Number of Cloud Satellites	18
Simulation time	30 (minutes)
Edge device counter time	20
Tasks generation rate	2
Network update interval	1 (second)
Earth radius	6,378,137 (meters)
Satellite min height	400,000 (meters)
Network bandwidth	1000 (Mbps)
Architectures	ALL
Orchestration algorithms	ROUND_ROBIN, TRADE_OFF, TRADI_POLLING, WEIGHT_GREEDY, ISOLINK

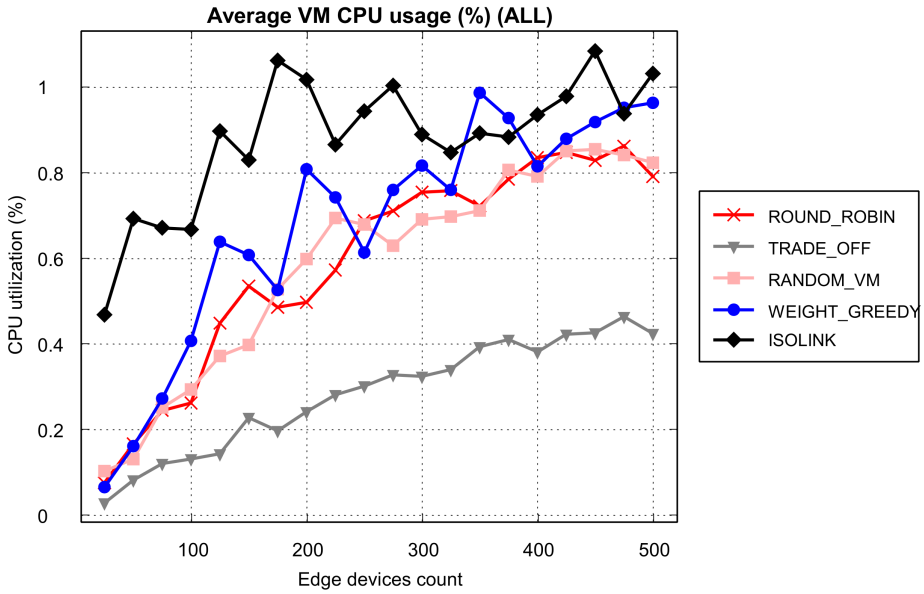


Figure V.1: Average VM CPU

3 Simulation Results

We exhibit and discuss the simulation results using several prominent performance metrics. Each metric provides insightful information regarding the behavior and performance of the IsoLink algorithm relative to competing orchestration approaches. The subsequent subsection discusses the results achieved during simulation demonstrations and includes both visual and numeric illustrations to facilitate an appraisal based on comparison:

The average VM CPU usage analysis, as shown in Figure V.1, showed that IsoLink used a large amount of CPU resources on every number of edge devices. This is in accordance with IsoLink's design to favor the execution of mist near to where they reside in order to reduce latency.

Comparatively, instances of Weight_Greedy also used relatively high amounts of CPU, although slightly lower than IsoLink. Conversely, orchestration approaches like Random_VM and Round_Robin sustained moderate levels of CPU usage. Trade_Off recorded the lowest average CPU usage, reflecting an overall more prudent strategy of allocating resources.

These findings indicate that although IsoLink is resource-consuming, its associated

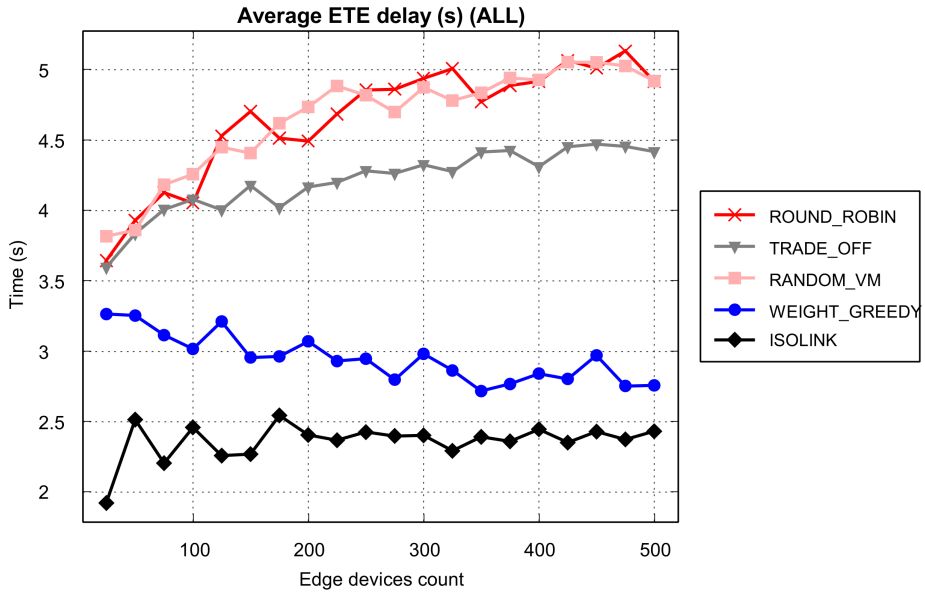


Figure V.2: The average end-to-end delay.

high CPU usage is an inevitable result of its latency-critical off-loading strategy and is thus appropriate for real-time and critical applications.

The End-to-End Delay analysis, as presented in Figure V.2, illustrates that IsoLink has a superior delay-minimizing capability compared to other task-offloading approaches. In all numbers of edge devices, IsoLink's ETE delay remained the lowest, below 2.5 seconds, even with an increase in edge devices.

This is an indication of IsoLink's design emphasis on mist-level task execution, which lowers its physical and network latency from task creation to execution. With an emphasis on low-latency communication and efficient task selection, IsoLink is able to achieve timely task completion.

Weight_Greedy follows IsoLink's performance, achieving a moderate and stable delay, while Random_VM, Round_Robin, and Trade_Off techniques have much higher delay times, typically higher than 4.5 seconds, that may be unacceptable in latency-critical applications.

The average energy consumption per data center in Figure V.3 shows that IsoLink consumed the least energy with respect to all offloading techniques. IsoLink's average energy consumption remains between 182–187 dBW with slight variations as

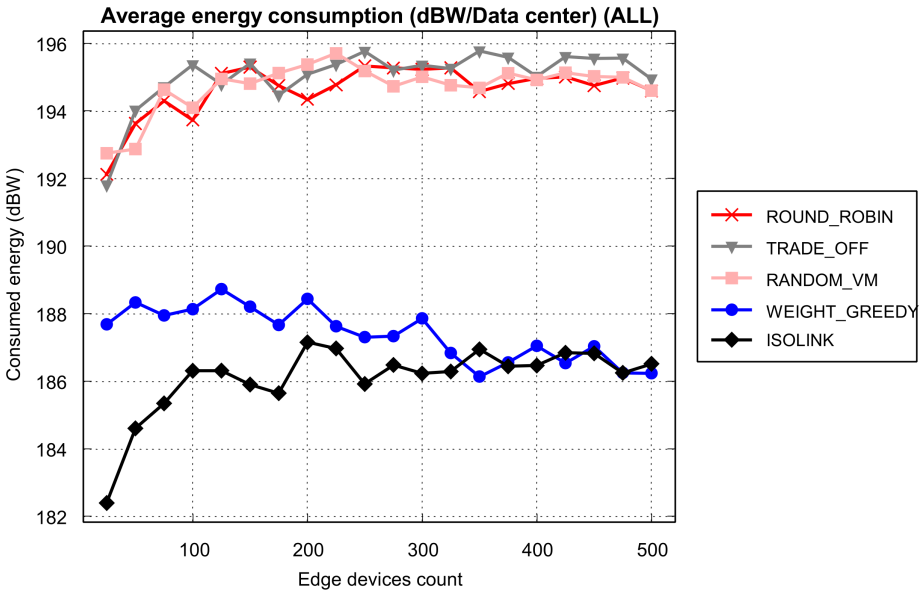


Figure V.3: The average energy consumption.

more edge devices.

Such consumption occurs from a highly effective scenario, as the offloading selected by the algorithm to the mist-layer VMs, which are proximate to the data sources, fosters lower processing and communication latency. Therefore, energy consumption is dedicated to performance and task completion instead of transmission.

In contrast, Round_Robin, Trade_Off, and Random_VM algorithms display an increase in energy consumption with no variation, averaging over 194 dBW. These techniques fail to appropriately offload tasks, either unaware of proximity or task partitioning.

Weight_Greedy is an improvement over those traditional approaches but still consumes more than IsoLink, averaging approximately 188 dBW.

The total network usage analysis of all layers, as shown in Figure V.4, shows that IsoLink is more efficient compared to other offloading strategies. Although an increasing network usage is observed in all algorithms as the number of edge devices increases, IsoLink has the lowest usage time in every case, never surpassing 17,000 seconds even for 500 devices, while other algorithms exceed 45,000 seconds.

This superior performance is due to IsoLink's mist-level task execution strategy,

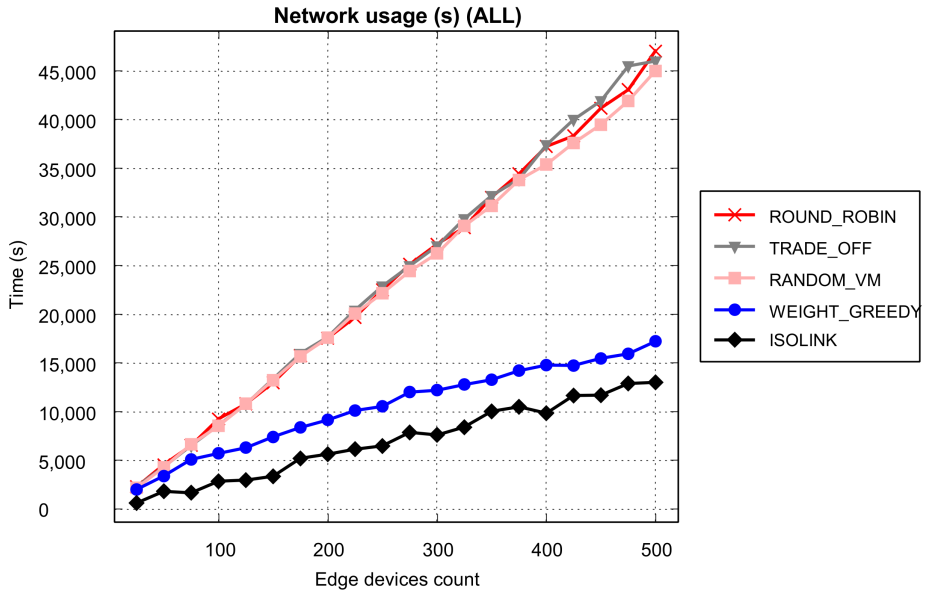


Figure V.4: The network usage.

which keeps transmission distance to a minimum and minimizes communication time. IsoLink prevents unnecessary multiple-layer routing by smart-classifying tasks and choosing the most appropriate VM based upon task type and context.

In contrast, it is observed that Round Robin, Trade_Off, and Random_VM exhibit maximum network utilization, often as a result of insufficient spatial or contextual insights in their allocation choices, resulting in remote VM choice, as well as an elevated transmission overhead.

Weight_Greedy provides moderate performance, surpassing conventional approaches but still trailing behind IsoLink, particularly as edge device numbers grow larger.

Successfully Executed Tasks The findings presented in Figure V.5 confirm IsoLink's task success rate superiority. For every number of edge devices that was tested, IsoLink has the highest success rate, at an average of about 99%, and shows outstanding stability as the network scales larger and larger.

Its high reliability is attributed to IsoLink's task classification based on context and its mechanism of dynamically selecting VM, which, by avoiding congested and overloaded nodes, is capable of ensuring task execution within deadlines, particularly

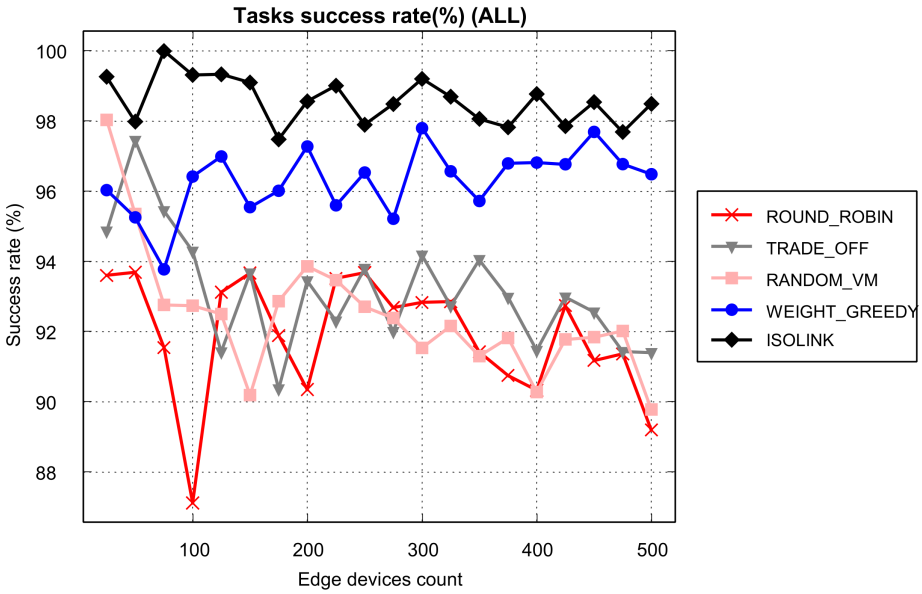
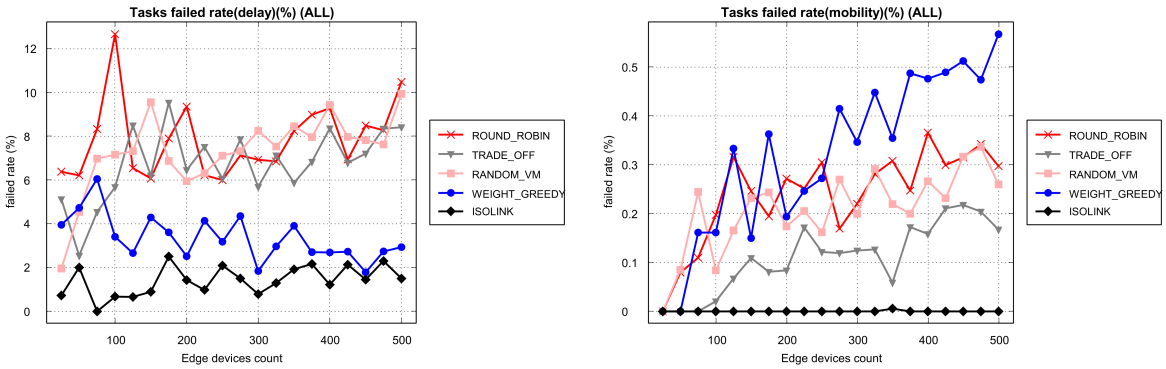


Figure V.5: Task success rate.

in dense edge environments.

Weight_Greedy, by contrast, also has relatively high success rates, of about 96-97% on average, which reflects moderate adaptability but is still worse than IsoLink's. Traditional methods like Random_VM, Trade_Off, and Round_Robin, on the other hand, have lower and unstable success rates, especially at high edge device loads. The instability is largely attributed to inefficient task scheduling and an excessive dependence on random or fixed policies, which, in turn, cause execution latency or failure.

Figure V.6a and V.6b illustrates **the task failure rate due to delay and mobility**, where the proposed strategy performs well in decreasing the task failure rate due to delay, thereby being an excellent option for delay-sensitive applications. The results in Figure V.6b indicate that the IsoLink algorithm has a 0% failure rate caused by moving across the network, performing much better than the other approaches in this aspect. The excellent performance is due to IsoLink's strategy of choosing the closest and most reliable mist-layer virtual machines from the context-aware offloading system. Avoiding communication with satellites or other rapidly moving networks, IsoLink greatly decreases the chances of connections failing during a task.



(a) Task failure due to delay

(b) Task failure due to mobility

Figure V.6: Task failure rate

It is also important that IsoLink's strategy uses the mist layer for enhanced stability and decreased distance. Because of this design, tasks are given to nodes that are reliable and rarely disconnect because they travel. Unlike Weight_Greedy, Random_VM, and Round_Robin, which do not take node mobility into account, this results in more task failures as mobility increases the risk of interrupted tasks during satellite transfer.

Therefore, the strong reliability and ease of use with IsoLink make it suitable for use in demanding real-time or critical applications in infrastructures with high dynamic demands.

Figure V.7 depicts the particular application categories as the number of satellites increases from 25 to 500. While each application exhibits a consistent growth trend, as the number of satellites increases, Smart Farming and IoT have the most tasks, while Augmented Reality and E-Health follow behind; Heavy Computing Applications have the least increase in task generation. This demonstrates the reliance on satellite systems that support future, low-latency applications, especially for real-world efforts such as smart farming and smart systems.

On Figure V.8, we see how the number of satellites affects the amount of different task types (Real-Time, Latency-Tolerant, Heavy). As more satellites are added, more than half of the tasks included fall into the latency-tolerant category, and the number of these tasks is rising rapidly. In the range, real-time tasks occur moderately frequently, while heavy tasks are used least. This means that many generated tasks can handle a small delay, which suits them for mist-layer use. The fewer heavy tasks seen in the

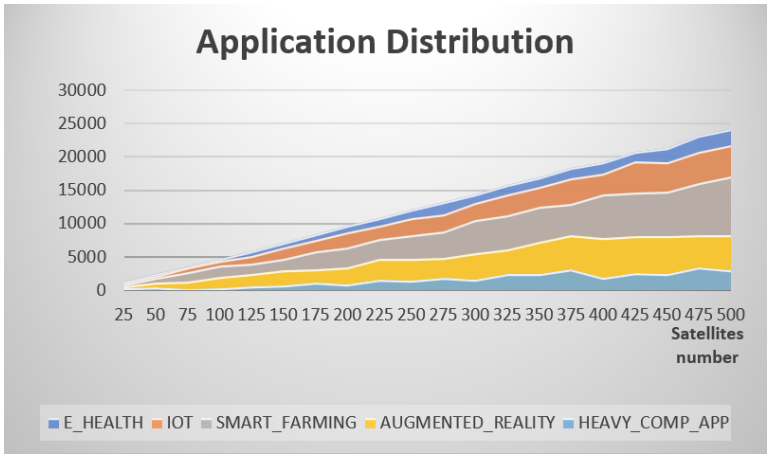


Figure V.7: Applications types distribution per number of satellites

mist layer confirm that it performs best when tasks are light and delays are acceptable, and it prefers to distribute tasks that do not require intensive computational resources, which aligns with the capabilities of the low-resource mist layer.

Table V.2 presents a comparison of the task offloading algorithms according to various performance evaluation metrics. The top-rated algorithm in each metric is highlighted in bold. The final line represents the weighted total score where the highest score received represents the most efficient algorithm overall. According to the table, IsoLink is the most efficient compared to all others, and basically succeeds in delay minimization and increasing success rates.

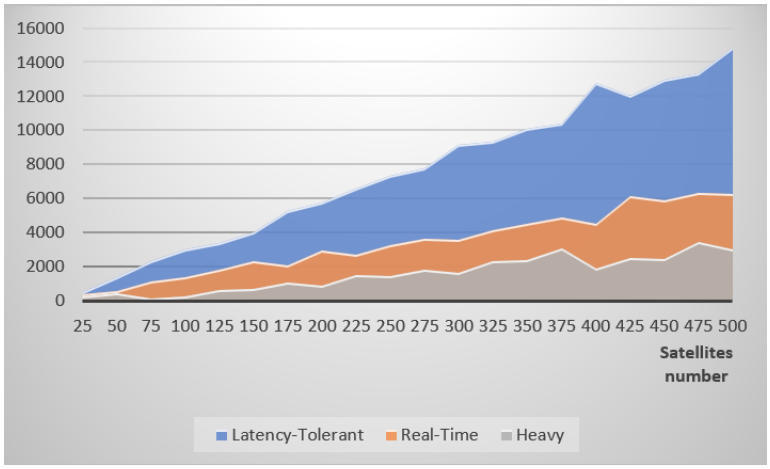


Figure V.8: IsoLink tasks classification per number of satellites

Table V.2: Comparison of Offloading Algorithms Based on Multiple Metrics

schemes	metrics	CPU Usage	End-to-End Delay	Energy Consumption	Network Usage	Task Success Rate	Task Failure due to Delay	Task Failure due to Mobility	Total score	Rank
	weights	0.10	0.20	0.15	0.15	0.20	0.10	0.10	1.00	
	IsoLink	1	**5**	**5**	**5**	**5**	**5**	**5**	**4.60**	**1**
	Weight_Greedy	2	4	4	4	4	4	1	3.50	2
	Random_VM	4	1	1	1	3	2	3	2.00	3
	Round_Robin	4	1	1	1	1	1	2	1.40	5
	Trade_Off	**5**	2	1	1	2	3	4	2.30	4

4 Summary

The chapter reviewed and evaluated the IsoLink offloading strategy when compared to four representative approaches, including Weight_Greedy, Round_Robin, Trade_Off, and Random_VM. By running detailed simulations, we examined how multiple key performance factors, such as CPU usage, overall latency, electric consumption, network usage, task success rate, and failures, changed with delays and mobility. IsoLink demonstrated better performance than the other approaches in delay-sensitive metrics, such as lowering the average end-to-end delay and raising the success rate of tasks. Due to its effective mist-level task allocation, IsoLink proved to be the most efficient from an energy and network utilization perspective, but its CPU usage was higher than that of the other architectures.

General Conclusion and Future Work

This thesis addresses one of the most critical and timely issues in distributed systems: reducing end-to-end latency in satellite-based mist computing systems, particularly to support delay-critical applications such as remote healthcare, autonomous systems, smart agriculture, and real-time environmental monitoring. These applications not only need real-time processing of the data but also highly resilient infrastructures that can provide low-latency services even in geographically distant and isolated areas. Cloud computing architectures have proven to fall short in these scenarios because the distance-aware delay significantly impacts the processing in traditional cloud models. Decentralized paradigms, including edge and mist computing, have emerged as the new trend in the computing paradigm to reduce the distance between the sources of the data and the corresponding computations. The inclusion of satellite infrastructures within these paradigms brings new opportunities but also new challenges in the management of tasks, connectivity, and orchestration.

To address these issues, we presented IsoLink, a context-aware and smart task offloading algorithm specifically for the mist layer of satellite-based edge computing. IsoLink is characterized by its novel multi-step approach in which incoming tasks are profiled by matching them with the respective application types. It further categorizes the tasks based on particular criteria such as tolerable execution delay, container size, needed computational capacity, and

complexity. IsoLink dynamically allocates weights to the key parameters, mainly the communication and execution delays based on the real-time constraints of the task. These parameters are then normalized over the entire set of candidate virtual machines, enabling the algorithm to derive a composite score for each VM and pick the one with the lowest score for the best execution of tasks.

We used the SatEdgeSim simulation platform, which was modified to accommodate multi-layer satellite topologies (cloud, edge, and mist) and varied realistic network and resource conditions, to implement and benchmark IsoLink. We evaluated the performance of IsoLink against a number of established offloading strategies, such as Weight_Greedy, Round_Robin, and Random_VM, via comprehensive simulations. We based the comparison on a number of performance metrics, including average CPU usage, end-to-end latency, energy expenditure, network usage, success rate of tasks, and failure rate of tasks caused by delay and mobility. The results were consistently in favor of IsoLink, as it performed better in satisfying the requirements of latency-bound applications. Of particular interest is the fact that IsoLink was able to maintain the average end-to-end delay below 2.5 seconds as the number of edge devices was varied, a result that was significantly better compared to its counterparts.

The power consumption analysis demonstrated that IsoLink utilized the lowest power by leveraging neighboring mist-layer satellites to execute tasks, thereby minimizing the overhead of transmission and processing. Additionally, it achieved an impressive success rate of over 99% for tasks, indicating its reliability and strength in resource-limited situations. Although IsoLink used a moderately high CPU, this was justified through its aggressive scheduling to achieve stringent delay constraints.

The design of the algorithm focuses on responsiveness and context-awareness, which are key to contemporary applications running in dynamic and usually unpredictable scenarios. In spite of these positives, however, our research also identified a number of areas where IsoLink can be improved.

Although the IsoLink algorithm was efficient in mist-layer offloading, some other directions can be researched to boost its usefulness:

- **Scalability Assessment:** Test IsoLink on large-scale scenarios handling many tasks and devices to measure its effectiveness.
- **Support for Task Segmentation Management:** Suggest making changes to IsoLink so that large tasks can be cut into pieces that can be run simultaneously in parallel on different mist nodes. This may significantly reduce latency and better manage resources in constrained satellite environments.
- **Security Consideration:** Apply light encryption, check user identity, and blockchain protocols to secure and dependable offloading of tasks.
- **Integrating AI-Module inside IsoLink:** To enable intelligent, adaptive virtual machine selection, incorporate machine learning models into IsoLink.
- **Handling satellite disconnection and handover** by proposing solutions for task-switching to other satellites.

In summary, IsoLink offers a comprehensive, intelligent model for offloading tasks that not only knows application-specific limitations but is also flexible to accommodate the intricacies of satellite-based systems. The innovations of this work include the development, deployment, and experimentation of a delay-conscious, energy-efficient, and context-aware task orchestration system

that fills the gap between the requirements of the IoT and the capabilities of the satellite. By establishing the grounds for subsequent research and development, this thesis illuminates avenues toward more adaptive, more intelligent, and more secure distributed systems. With scalability assessment, support for task segmentation management, security consideration, and integrating AI, IsoLink has the potential to become an anchor component in the orchestration layers of next-generation satellite-edge systems.

Appendices

1 Classify And Match Application

Algorithm 2: classifyAndMatchApplication

Input: Task `task`, List of Applications `applications`

Output: Matched application name

```
1  function classifyAndMatchApplication(task, applications);
2  if task == null then
3  |   return "unknown";
4  end
5  Extract task parameters:;
6      maxDelay ← task.maxDelay;
7      containerSize ← task.containerSize;
8      requiredCore ← task.requiredCore;
9      taskLength ← task.taskLength;
10 bestScore ← ∞;
11 matchedApplicationName ← "No_Match";
12 foreach app in applications do
13 |   score ← |app.maxDelay − maxDelay|
14 |           +|app.containerSize − containerSize|/1000
15 |           +|app.requiredCore − requiredCore| × 10
16 |           +|app.taskLength − taskLength|/10000;
17 |   if score < bestScore then
18 |   |   bestScore ← score;
19 |   |   matchedApplicationName ← app.name;
20 |   end
21 end
22 return matchedApplicationName;
23 end function
```

2 Classify Task Based On Properties

Algorithm 3: classifyTaskBasedOnProperties

Input: maxDelayValue, containerSizeValue, requiredCoreValue, taskLength

Output: Task Type (real-time, latency-tolerant, or heavy)

```
1  function classifyTaskBasedOnProperties(maxDelayValue,
     containerSizeValue, requiredCoreValue, taskLength);
2  SMALL_CONTAINER_THRESHOLD ← 30000;
3  MEDIUM_CONTAINER_THRESHOLD ← 60000;
4  SMALL_TASK_LENGTH ← 100000;
5  if maxDelayValue ≤ 10 and containerSizeValue ≤
     SMALL_CONTAINER_THRESHOLD and requiredCoreValue < 2 and
     taskLength ≤ SMALL_TASK_LENGTH then
6  |  return "real-time";
7  end
8  else if maxDelayValue ≤ 30 and containerSizeValue ≤
     MEDIUM_CONTAINER_THRESHOLD then
9  |  return "latency-tolerant";
10 end
11 else
12 |  return "heavy";
13 end
14 end function
```

3 Read Applications From XML

Algorithm 4: readApplicationsFromXML

Output: List of Application objects

```
1  function readApplicationsFromXML();
2  applications ← [ ] ;                                  // Initialize empty list
3  try ;
4      Load XML file: "SatEdgeSim/settings/applications.xml";
5      Parse and normalize the XML document;
6      Get the list of <application> elements;
7      foreach applicationElement in list do
8          Extract attributes:;
9              name, maxDelay, containerSize, requestSize,;
10             resultsSize, taskLength, requiredCore, poissonInterarrival;
11         taskType ← classifyTaskBasedOnProperties(maxDelay,
            containerSize, requiredCore, taskLength);
12         app ← new Application(name, maxDelay, containerSize, requestSize,
            resultsSize, taskLength, requiredCore, poissonInterarrival, taskType);
13         Add app to applications list;
14     end
15 catch any exception:;
16     Print error message;
17 return applications;
18 end function
```

4 standardization

Algorithm 5: standardization

Input: List of double values `Pre_standar`

Output: List of normalized values `standard`

```
1  function standardization(Pre_standar);
2  standard ← [ ] ;                                  // Initialize empty list
3  premax ← max(Pre_standar);
4  premin ← min(Pre_standar);
5  foreach value in Pre_standar do
6  |   temp ← (value − premin) / (premax − premin);
7  |   Append temp to standard;
8  end
9  return standard;
10 end function
```

Bibliography

[1] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. (SP 800-145), 2011. URL https://doi.org/10.6028/NIST.SP.800-145.

[2] Salam Al-E'mari, Yousef Sanjalawe, Ahmad Al-Daraiseh, Mohammad Bany Taha, and Mohammad Aladaileh. Cloud Datacenter Selection Using Service Broker Policies: A Survey. *Computer Modeling in Engineering & Sciences*, 139(1):1–41, 2024. ISSN 1526-1506. doi: 10.32604/cmes.2023.043627. URL https://www.techscience.com/CMES/v139n1/55130.

[3] A. Fu. 7 different types of cloud computing structures. *UniPrint.net*, October 23 2022. URL https://www.uniprint.net/en/7-types-cloud-computing-structures/. Accessed: 2025-02-09.

[4] Madhusanka Liyanage, Ijaz Ahmad, Ahmed Bux Abro, Andrei Gurtov, and Mika Ylianttila, editors. *A Comprehensive Guide to 5G Security*. Wiley, 1 edition, March 2018. ISBN 978-1-119-29304-0 978-1-119-29307-1. doi: 10.1002/9781119293071. URL https://onlinelibrary.wiley.com/doi/book/10.1002/9781119293071.

[5] Juha-Jaakko Heiskari. COMPUTING PARADIGMS FOR RESEARCH: CLOUD VS. EDGE. 2022.

[6] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, April 2010. ISSN 0001-0782, 1557-7317. doi: 10.1145/1721654.1721672. URL https://dl.acm.org/doi/10.1145/1721654.1721672. Number: 4.

[7] John W. Rittinghouse and James F. Ransome. *Cloud Computing: Implementation, Management, and Security*. CRC Press, 1 edition, March 2017. ISBN 978-1-315-11021-9. doi: 10.1201/9781439806814. URL https://www.taylorfrancis.com/books/9781439806814.

[8] S. Kelkar. Challenges and opportunities with cloud computing. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(4):4567–4573, 2015. ISSN 2320-9801. URL https://d1wqtxts1xzle7.cloudfront.net/81913447/challenges-and-opportunities-with-cloudcomputing-libre.pdf. Accessed: 2025-02-09.

[9] K. Paulsen. The challenges of cloud computing in 2025. *TV Technology*. URL

https://www.tvtechnology.com/opinion/the-challenges-of-cloud-computing-in-2025. Accessed: 2025-02-09.

[10] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of Cloud computing and Internet of Things: A survey. *Future Generation Computer Systems*, 56: 684–700, March 2016. ISSN 0167739X. doi: 10.1016/j.future.2015.09.021. URL https://linkinghub.elsevier.com/retrieve/pii/S0167739X15003015.

[11] A. Gupta and R. K. Jha. A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access*, 3:1206–1232, 2015. ISSN 2169-3536. doi: 10.1109/ACCESS.2015.2461602. URL http://ieeexplore.ieee.org/document/7169508/.

[12] Mahadev Satyanarayanan. The Emergence of Edge Computing. *Computer*, 50(1):30–39, January 2017. ISSN 0018-9162. doi: 10.1109/MC.2017.9. URL http://ieeexplore.ieee.org/document/7807196/.

[13] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, October 2016. ISSN 2327-4662. doi: 10.1109/JIOT.2016.2579198. URL http://ieeexplore.ieee.org/document/7488250/.

[14] R. Arunkumar, C. Dhivya, M. M. Asik, and V. Balamurugan. Farm automation and iot technologies in agriculture. In *Advances in Agricultural Sciences*, pages 356–381. Eleyon Publishers - Royal Book Publishing International, November 2024.

[15] Koustabh Dolui and Soumya Kanti Datta. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6, Geneva, Switzerland, June 2017. IEEE. ISBN 978-1-5090-5873-0. doi: 10.1109/GIOTS.2017.8016213. URL http://ieeexplore.ieee.org/document/8016213/.

[16] Mung Chiang and Tao Zhang. Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, December 2016. ISSN 2327-4662. doi: 10.1109/JIOT.2016.2584538. URL http://ieeexplore.ieee.org/document/7498684/.

[17] Pedro Juan Roig, Salvador Alcaraz, Katja Gilly, Cristina Bernad, and Carlos Juiz. Modeling of a Generic Edge Computing Application Design. *Sensors*, 21(21):7276, November 2021. ISSN 1424-8220. doi: 10.3390/s21217276. URL https://www.mdpi.com/1424-8220/21/21/7276.

[18] Fatema Vhora and Jay Gandhi. A Comprehensive Survey on Mobile Edge Computing: Challenges, Tools, Applications. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 49–55, Erode, India, March 2020. IEEE. ISBN 978-1-7281-4889-2. doi: 10.1109/ICCMC48092.2020.ICCMC-0009. URL https://ieeexplore.ieee.org/document/9076528/.

[19] Santosh Kumar Yadav and Rakesh Kumar. EVACON-Rainsnow Computing: An Amalgamation of Cloud and Its Inherited Computing to Encourage End User for Both Localized and Globalized Remote Computing. *Wireless Personal Communications*, 132(4):

2737–2792, October 2023. ISSN 0929-6212, 1572-834X. doi: 10.1007/s11277-023-10741-5. URL https://link.springer.com/10.1007/s11277-023-10741-5.

[20] Shams Forruque Ahmed, Shanjana Shuravi, Shaila Afrin, Sabiha Jannat Rafa, Mahfara Hoque, and Amir H. Gandomi. The Power of Internet of Things (IoT): Connecting the Dots with Cloud, Edge, and Fog Computing, 2023. URL https://arxiv.org/abs/2309.03420. Version Number: 1.

[21] Wikipedia contributors. Industrial internet of things. *Wikipedia, The Free Encyclopedia*, December 17 2024. URL https://en.wikipedia.org/wiki/Industrial_internet_of_things. Accessed: 2025-02-09.

[22] H. Harishini. Unveiling the future of cloud computing. Medium, October 22 2023. URL https://medium.com/@harishini2002/unveiling-the-future-of-cloud-computing-4114e128c45c. [Online]. Accessed: 2025-02-09.

[23] Acropolium. Cloud computing in healthcare: Benefits, challenges, and applications. Acropolium Blog, 2025. URL https://acropolium.com/blog/cloud-computing-healthcare/. [Online]. Accessed: Feb. 7, 2025.

[24] R. Morabito. Lightweight virtualization in edge computing for internet of things. Master's thesis, Aalto University, Dept. of Communications and Networking, Espoo, Finland, April 2019. URL https://aaltodoc.aalto.fi/items/f35a4cfa-135a-44f2-9b64-1d1156df2b8e. Accessed: 2025-02-09.

[25] O. Andrieiev. Edge computing definition and use cases. Jelvix Blog, 2025. URL https://jelvix.com/blog/what-is-edge-computing. [Online]. Accessed: Feb. 7, 2025.

[26] Junyong Wei, Suzhi Cao, Siyan Pan, Jiarong Han, Lei Yan, and Lei Zhang. SatEdgeSim: A Toolkit for Modeling and Simulation of Performance Evaluation in Satellite Edge Computing Environments. In *2020 12th International Conference on Communication Software and Networks (ICCSN)*, pages 307–313, Chongqing, China, June 2020. IEEE. ISBN 978-1-7281-9815-6. doi: 10.1109/ICCSN49894.2020.9139057. URL https://ieeexplore.ieee.org/document/9139057/.

[27] Feng Wang, Dingde Jiang, Sheng Qi, Chen Qiao, and Lei Shi. A dynamic resource scheduling scheme in edge computing satellite networks. *Mobile Networks and Applications*, 26:597–608, 2021.

[28] Jiaxin Zhang, Xing Zhang, Peng Wang, Liangjingrong Liu, and Yuanjun Wang. Double-edge intelligent integrated satellite terrestrial networks. *China Communications*, 17(9):128–146, 2020.

[29] Yuxuan Wang, Jun Yang, Xiye Guo, and Zhi Qu. Satellite edge computing for the internet of things in aerospace. *Sensors*, 19(20):4375, 2019.

[30] Xiangqiang Gao, Rongke Liu, and Aryan Kaushik. Virtual network function placement in

satellite edge computing with a potential game approach. *IEEE Transactions on Network and Service Management*, 19(2):1243–1259, 2022.

[31] Trong-An Bui, Pei-Jun Lee, Chun-Sheng Liang, Pei-Hsiang Hsu, Shiuan-Hal Shiu, and Chen-Kai Tsai. Edge-computing-enabled deep learning approach for low-light satellite image enhancement. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 17:4071–4083, 2024.

[32] Yuxuan Wang, Jun Yang, Xiye Guo, and Zhi Qu. A game-theoretic approach to computation offloading in satellite edge computing. *IEEE Access*, 8:12510–12520, 2019.

[33] Changfeng Ding, Jun-Bo Wang, Ming Cheng, Min Lin, and Julian Cheng. Dynamic transmission and computation resource optimization for dense leo satellite assisted mobile-edge computing. *IEEE Transactions on Communications*, 71(5):3087–3102, 2023.

[34] Israel Leyva-Mayorga, Marc Martinez-Gost, Marco Moretti, Ana Pérez-Neira, Miguel Ángel Vázquez, Petar Popovski, and Beatriz Soret. Satellite edge computing for real-time and very-high resolution earth observation. *IEEE Transactions on Communications*, 71(10): 6180–6194, 2023.

[35] Messaoud Babaghayou, Noureddine Chaib, Leandros A Maglaras, Yagmur Yigit, Mohamed Amine Ferrag, Carol Marsh, and Naghmeh Moradpoor. Apollo: a proximity-oriented, low-layer orchestration algorithm for resources optimization in mist computing. *Wireless Networks*, pages 1–16, 2024.

[36] Messaoud Babaghayou, Noureddine Chaib, Leandros Maglaras, Yagmur Yigit, Mohamed Amine Ferrag, and Carol Marsh. Proximity-driven, load-balancing task offloading algorithm for enhanced performance in satellite-enabled mist computing. In *International Wireless Internet Conference*, pages 29–44. Springer, 2023.