**Optimizing DCNSFog with SDN Integration and Load Balancing Algorithms for Enhanced Performance in Fog Computing**

Design and Analysis of Algorithms REPORT

***Submitted by***

## BL.EN.U4CSE22032 Kandibanda Lohith

## BL.EN.U4CSE22034 Lakshmi Priya P

## BL.EN.U4CSE22028 Joel M Johnson

***in partial fulfillment for the award of the degree of***

# BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING



AMRITA SCHOOL OF ENGINEERING, BANGALORE AMRITA VISHWA VIDYAPEETHAM **BANGALORE 560 035**

# ABSTRACT

The arising paradigm of fog computing emerged to counter the drawback of traditional cloud computing, especially in terms of latency-sensitive and bandwidth-intensive applications within the Internet of Things (IoT). As IoT networks grow complex, challenges include resource management, execution efficiency, and scalability. The system considers the challenges of latency, resource management, and scalability that occur in large-scale Internet of Things (IoT) networks with the proposal of a Distributed Cloud Networking System based on fog computing. This study used the iFogSim simulator in order to compare the performance of the two different DCNS configurations between a basic fog architecture and an advanced configuration with the integration of Software-Defined Networking (SDN) and intelligent load-balancing techniques. The benchmarking results show a significant reduction in key performance metrics, which also indicates 16% reduction in execution time, 20% savings in execution cost, and better resource optimization. SDN integration would achieve dynamic routing and task scheduling and improve network efficiency, throughput, and response time considerably. This research has shown potential SDN-enhanced fog computing architectures on the optimization of IoT networks and scalable and efficient solutions for modern distributed computing environments.

**KEYWORDS:**  Fog Computing, Distributed Cloud Networking, Software-Defined Networking (SDN), iFogSim Simulation, Load Balancing

|  |
| --- |
| **TABLE OF CONTENTS** |
| **Topic** |
| Abstract |
|  |
| 1. INTRODUCTION |
| 1 Introduction |
| 1.2 Motivation |
| 1.3 Problem Statement |
| 2. LITERATURE SURVEY |
| 2.1 Related Work |
| 2.2 Summary of Literature Survey |
| 3. SYSTEM MODEL AND PROBLEM FORMULATION  3.1 System Overview  3.2 System Architecture |
| 3.2.1 Functionality of DCNSFog |
| 3.2.2 Performance evaluation metrics |
| 3.2.3 Baseline evaluation  3.3 Integration of SDN and Hybrid Load Balancing  3.3.1 Dynamic routing with SDN  3.3.2 Hybrid Load Balancing  3.4 Methodology Overview |
|  |
|  |
| 4. PERFORMANCE EVALUATION OF DCNSFOG  4.1 Comparitive Analysis  4.1.1 Evaluation of Baseline DCNSFog  4.1.2 Evaluation of Optimized DCNSFog  4.2 Observation  4.3 Result and Analysis |
| 5. CONCLUSION |
| 6. FUTURE SCOPE |

# INTRODUCTION

### Introduction

The Internet of Things is reinventing health care and smart cities by bringing widespread communication and real-time analysis to sensors and devices across the world. However, it has its set of difficulties in large-scale deployments in terms of network performance, resource management, and latency. Traditional cloud computing paradigms, although useful for most of the applications, cannot fulfill the requirements of IoT systems due to issues such as latency, bandwidth, and data processing centrally. This leads to delays, inefficiencies, and an inability to support large-scale real-time applications. These challenges are particularly critical in fields such as healthcare, where timely decision-making spells the difference between life and death, and smart cities, where real-time data processing paves the way for managing critical infrastructure[1].

Fog computing has emerged as a promising solution to these challenges by bringing computation and data storage closer to the edge of the network, reducing latency and bandwidth consumption. This approach allows more efficient data processing and decision-making in distributed devices, especially in IoT networks. Nevertheless, despite its flexibility, significant issues in resource management, fault tolerance, and scalability prevent fog computing from achieving widespread adoption. This work introduces an innovative Distributed Cloud Networking System, which combines SDN and intelligent load balancing techniques to address such issues. The project will utilize the widely recognized simulation tool, iFogSim, to compare basic fog computing configurations with advanced configurations enabled by SDN, hence showing improvements in some of the key performance metrics to include execution time, energy consumption, and network usage[2][5].

Through this research, the study seeks to provide actionable insights into the optimization of IoT networks by exploring the synergies between SDN and fog computing. By enhancing network management and load distribution, this work proposes a solution to the scalability and efficiency challenges facing modern IoT applications, particularly in high-demand environments such as smart cities and healthcare[6][7].

### Motivation

The motivation behind this project arises from the huge volumes of data generated at the network edge by increasingly large-scale and complex Internet of Things applications. This calls for much more efficient and responsive network architectures than traditional cloud computing, with its foundation in centralized data centers, to meet the real-time processing requirements of those huge applications and thus avoid latency and congestion as well as inefficiency. The computation and storage of data closer to the network edge, close to the consumers, would definitely reduce these problems with the promise of fog computing. However, resource management, scalability, and load balancing are some of the challenges that prevent the realization of the full potential of fog computing networks in IoT networks. With this in mind, the project integrates Software-Defined Networking (SDN) and advanced techniques for load balancing in a Distributed Cloud Networking System (DCNS). Since the use of iFogSim can model and evaluate both traditional and advanced DCNS configurations, focusing on key performance metrics such as execution time, energy consumption, and network usage.

### Problem Statement

The issue however lies in the fact that the data-centric networking system incorporating fog computing, the lack of centralized control leads to inefficient use of resources and suboptimal latency particularly in scenarios with high demand. With an increase in IoT complexity, it becomes a challenge in terms of managing resources and fulfilling real-time processing demands. Furthermore, this inefficiency leads to network congestion without dynamic resource allocation. This research attempts to address these challenges by embedding SDN in the DCNSFog model in order to achieve centralized, real-time resource management control. Thus, the proposed solution aims at making an improved workload distribution across the fog and cloud layers using an adaptive load-balancing algorithm. The improved model would be tested iteratively and its key metrics, such as latency, throughput, and resource utilization, checked to serve as proof that SDN can optimize a fog computing environment with regard to scalability and responsiveness.SDN’s effectiveness in optimizing fog computing environments for more responsive and scalable data processing

# LITERATURE SURVEY

### Related Work

The evolution of IoT applications has made the networks much more complex and more scaled up. Traditional centralized data centers-based cloud computing fails to efficiently process the gigantic volume of data at the edge of IoT networks. With the growing need for real-time processing of the increasing data, the centralized model of cloud systems limits the output, characterized by higher latency and network congestion. This has led to the emergence of Fog Computing as an alternative architecture. Fog Computing, by distributing computational tasks closer to the IoT devices at the network edge, addresses these limitations by reducing latency and bandwidth consumption. However, the integration of Fog Computing within large-scale IoT ecosystems brings its own set of challenges related to resource management, load balancing, and scalability. These issues need to be addressed to fully exploit the potential of Fog Computing for IoT applications [1].

Fog Computing literature identifies a number of approaches aimed at addressing the inherent complexities of centralised cloud systems. Several research studies have discussed integrating Fog Computing with Cloud Computing as an efficient approach to facilitate even better data processing and storage across distributed network nodes. The notion behind distributing data processing and storage across multiple layers in the Edge-Fog-Cloud Continuum is that the computing tasks are dynamically profiled to decide where they can be processed efficiently on IoT devices, given that sometimes these devices need to offload processing to the cloud for more resource-intensive processing. Such an architecture addresses problematic issues like latency, network congestion, and energy consumption, thereby making real-time IoT applications feasible for the monitoring of health and smart cities. Despite these developments, various research studies have revealed that there are still large voids left in the achievement of flawless interoperability across the different layers of the Edge-Fog-Cloud architecture, particularly in heterogeneous IoT devices [2].

Another seriously underexplored area is that of Fog Computing system scalability. Many frameworks have been proposed in the literature to scale up cloud-based IoT systems, but relatively few solutions manage to abate the involved complexities of the Fog environment. Fog Computing nodes are spread geographically and usually resource-constrained, which makes it hard to manage the dynamic workload. As the scale of IoT networks increases, these problems become more critical, and developing systems that can dynamically distribute resources and balance loads across Fog nodes becomes inevitable. Moreover, the heterogeneous amount of devices involved in a Fog network makes it challenging to offer consistent performance in the system. Probably, more work is still to be done concerning the development of models that can efficiently accommodate large-scale Fog-IoT systems in view of its scalability concerns [3].

Apart from scalability, security is another paramount concern in the context of Fog Computing. The increased adoption of IoT in sensitive domains like healthcare, smart cities, and industrial monitoring makes data security and privacy in Fog networks a growing concern. Several studies indicate the need for secure Fog Computing environments, but many frameworks limit themselves to traditional cloud security models without discussing the unique decentralized challenges of Fog environments. Fog systems by their distributed and diversified nature comprise a wide array of devices with diverse levels of computational power and different security needs. The diversity makes it cumbersome to implement robust security mechanisms, and much research is still needed in the design of security frameworks to safeguard data transmissions between IoT devices and Fog nodes. The more devices are connected, the higher the need to ensure secure data handling and prevent unauthorized access to sensitive information in Fog environments is becoming [4].

While one promising solution to enhance Fog Computing environments is SDN, it facilitates centralized control of network resources and their efficient allocation across the distributed nodes. Static routing and management protocols used in traditional network architecture always fail in meeting the needs of high-speed demands of real Fog networks, resulting in inefficiency and congestion. Deploying SDN in a given network allows for dynamic routing of its traffic with the current condition of the network, thus improving performance. Recent studies have proven that the implementation of SDN could help optimize network traffic in the Fog Computing environment. However, there are challenges in the real-time adaptation of systems with an SDN, notably when deployed in dynamic IoT environments because the traffic patterns may change rapidly. Though significant improvements have resulted from the application of SDN in network performance, real-time adaptation and more specifically integration with Fog Computing remains a very active area of study [5][6].

Further benefits for the application of SDN in Fog Computing result from its use in dynamic load balancing techniques. Load balancing is an essential procedure to assure equilibrium, hence an acceptable load distribution across the nodes, so that no node is supposed to be overloaded with traffic. Dynamic load balancing algorithms based on Software-Defined Networking have been reported to significantly improve Fog network performance by optimizing task distribution and alleviating congestion within the network. These algorithms can adapt to changing network conditions and task loads in real-time, ensuring that resources are utilized efficiently. However, many existing studies focus only on theoretical models or simulations, with little attention paid to real-world implementation challenges. Additionally, the gap still exists in issues like fault tolerance, since dynamic load balancing algorithms have to consider node failures and ensure the workloads will be redistributed on operational nodes with little-to-no performance interference of the network [7].

Another significant fact in enhancing Fog Computing systems is fault tolerance. In distributed Fog networks, node failures are inevitable due to the nature of the architecture, which involves numerous geographically dispersed and heterogeneous devices. Most existing frameworks for Fog Computing fail to adequately address fault tolerance, which can lead to significant performance degradation when a node fails. Several studies have highlighted the importance of building fault-tolerant Fog networks, but this area still lacks comprehensive solutions. In particular, research on how to handle failures in real-time and reroute tasks to available resources is limited. Fault tolerance mechanisms must be integrated into Fog systems to ensure that workloads are continuously processed even in the event of a node failure. This involves designing algorithms that can detect and respond to failures quickly, ensuring minimal disruption to the overall system [8].

Besides the technical difficulties of resource management, load balancing, and fault tolerance, improvement in algorithms is important for optimizing performance across Fog networks. Several load balancing methods that exist today depend on static approaches that could not work well for dynamically changing IoT networks. Recently, improvements have been shown with dynamic load balancing, especially with SDN, improving network performance significantly. However, these algorithms still seem to face some problems in handling large-scale deployments, which include the involvement of many nodes and devices. Further, most research studies tend to avoid crucial issues such as energy efficiency as well as balancing energy consumption against performance in the network. With increasing IoT networks, energy-efficient algorithms will be required to ensure Fog Computing systems are sustainable over time as they remain performant [9][10].

A big gap in the literature still exists regarding the real-world applicability of Fog Computing in IoT environments. Most studies present theoretical models or simulation results, and little practical case study illustrates the effects of Fog Computing systems in real-world IoT applications. For instance, one of the areas in IoT-based video surveillance systems wherein Fog Computing improves performance by reducing the latency of cloud-based processing. However, issues related to resource provisioning, fault tolerance, and security are still unsolved in these realistic depictions. Additional case studies are required to test the theoretical models and elucidate the potential advantages of Fog Computing in various applications [11][12].

With SDN and Fog Computing, new opportunities for enriching the scalability, flexibility, and performance of IoT networks are presented. As noted in several studies, there is a lack of scalability that is required for big-scale deployment of IoT. Real-time adaptability and dynamic task offloading are needed to ensure that the network manages the changing demands and conditions within the network. Although SDN offers centralized control, the difficulty in integrating SDN with Fog Computing systems requires careful attention in designing the control and data planes such that there are seamless operations across various distributed Fog nodes [13][14].

Finally, the questionnaire highlights the challenges of QoS and energy efficiency of Fog Computing systems. QoS plays an important role in IoT applications. It requires real-time processing and low-latency response times for a system. Most of the current studies improved latency and throughput but did not appropriately address the trade-off between energy consumption and network performance. To achieve sustainable and efficient Fog Computing solutions, this factor will play a challenging role in the expansion of the IoT networks [5].

### Summary of Literature Survey

The Fog Computing integration with IoT networks promises to overcome the limitation and constraints of Cloud Computing, especially in the concern towards latency and bandwidth. Data processing near the IoT devices is made possible through Fog Computing; this eliminates network congestion while managing and handling real-time data, but then again, problems concerning resource management, scalability, and security appear. Several studies have looked at how Fog Computing could support Cloud Computing: laying down the continuum with Edge, Fog and Cloud layers to optimize the processing of data at every layer, then enhancing performance in IoT. Still, there are significant gaps in terms of complete interoperability of IoT devices and large-scale deployment [1-3].

The security of Fog Computing is particularly paramount because the application of IoT is spreading across critical sectors, particularly healthcare. Existing works tend to be cloud-centric security models and fail to consider the local challenges posed by the distributed architecture of Fog, such as heterogeneity in capabilities of nodes and secure data transfer among Fog nodes [4]. Moreover, SDN and Fog Computing integration has so far shown promise in improving network management through dynamic load balancing and real-time allocation of resources. SDN's centralized control over network resources helps address network congestion and ensures efficient data flow, but real-time adaptability and fault tolerance remain significant challenges [5-7].

In addition, the necessity of providing the load balancing mechanisms efficiently is highlighted with the recent focus on dynamic approaches to distribute network traffic. Although the SDN-based dynamic load balancing algorithms show improvement in performance, there is still a scope for improvements in aspects such as fault tolerance and energy efficiency [8][9]. Many of the studies are still mainly theoretical, without any practical case studies that realize the usability of the Fog Computing technology in various IoT contexts [10][11].

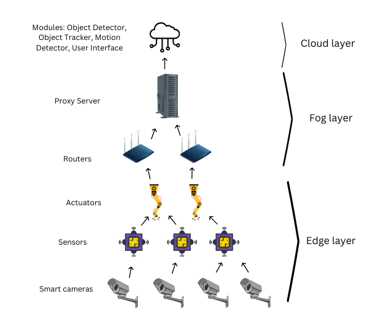
Summarily, as presented in relation to the challenges of Fog Computing in IoT ecosystems, large improvements have been achieved, but there is still a lot that needs to be filled concerning scalability, security, fault tolerance, and real-world implementation. Further research is therefore needed in developing comprehensive models which integrate SDN, improve energy efficiency, and ensure secure, reliable Fog-based IoT applications [12-15]..

1. **SYSTEM MODEL AND PROBLEM FORMULATION**

**3.1 System Overview**

The rapid growth of the Internet of Things (IoT) has given rise to data-intensive applications that demand real-time processing, low latency, and efficient resource utilization. One such application is intelligent surveillance, which involves tasks such as motion detection, object tracking, and video processing. Traditional cloud-only architectures face significant challenges, such as high latency and network congestion, when processing data generated by geographically distributed IoT devices. To overcome these limitations, the **Dynamic Cloud-Network Surveillance using Fog (DCNSFog)** system leverages the principles of **fog computing** to distribute computational and network resources closer to data sources.

**3.2 System Architecture**

****

**Fig. 1.** System Architecture

The DCNSFog architecture consists of three primary layers:

**Edge Layer:**

Comprised of smart cameras and IoT devices that capture real-time data, such as video feeds or motion alerts.

Performs lightweight processing tasks, including motion detection, at the edge to reduce the burden on upstream layers.

**Fog Layer:**

Includes intermediate devices such as routers, proxy servers, and regional fog nodes. These devices serve as processing hubs for tasks requiring moderate computation and local decision-making.

Fog nodes act as a bridge between the edge and the cloud layers, helping to minimize latency and reduce the load on cloud servers.

**Cloud Layer:**

Contains high-capacity servers capable of performing computationally intensive tasks such as object recognition, complex data analysis, and global decision-making.

Acts as the central repository for long-term storage and coordination.

### Functionality of DCNSFog

The system processes video streams and alerts generated by edge devices through the following workflow:

**Motion Detection:** IoT cameras analyze live video streams for motion events.

**Object Detection:** Detected motion triggers an alert, prompting fog or cloud nodes to analyze the video frames to identify objects.

**Object Tracking:** The identified objects are tracked across successive video frames to detect anomalies.

**User Interface:** Results are sent to a cloud-based user interface, allowing administrators to monitor events in real-time.

**3.2.2 Performance Evaluation Metrics**

To ensure the effectiveness of the DCNSFog system, we evaluate its performance in terms of the following metrics:

**Latency:** The end-to-end delay experienced during task execution, including data transmission, processing, and result delivery.

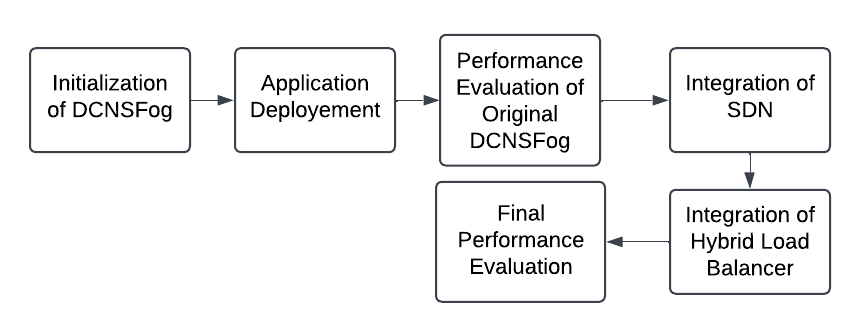
**Throughput:** The rate at which tasks are processed successfully within a given timeframe.

**Energy Consumption:** The total energy used by devices at each layer during task execution.

**Execution Time:** The total time required to complete a workflow (e.g., motion detection to result delivery).

**3.2.3 Baseline Evaluation**

The baseline system evaluates DCNSFog without advanced optimizations, such as dynamic routing or load balancing. This setup uses static routing between devices and relies on default module placement policies (e.g., ModulePlacementEdgewards in iFogSim). The results from this evaluation serve as the foundation for subsequent enhancements.



**Fig 2 : Flow of the project**

* 1. **Integration of SDN and Hybrid Load Balancing**

**3.3.1 Dynamic Routing with SDN**

Traditional fog computing architectures rely on static routing paths, which may lead to suboptimal performance under varying network conditions. To address this limitation, we integrate **Software-Defined Networking (SDN)** into DCNSFog. The SDN controller:

* Dynamically manages routing paths between devices based on real-time network conditions.
* Monitors device utilization, bandwidth, and latency to select the optimal path for task transmission.
* Enables centralized control and global visibility across the network, improving adaptability and scalability.

The SDN controller is implemented using a custom Java class within the iFogSim framework. It uses APIs to monitor network states and update routing tables dynamically. Tasks originating from edge devices are routed through the optimal path to fog or cloud nodes, minimizing latency and avoiding network congestion.

**3.3.2 Hybrid Load Balancing**

Load imbalance is a common issue in distributed systems, leading to overloading of certain devices and underutilization of others. To optimize resource utilization, we implement a Hybrid Load Balancer that combines:

**Round-Robin:** Distributes tasks in a circular order across available devices, ensuring uniform task allocation.

Round Robin is a simple and widely used load balancing algorithm that distributes tasks sequentially across a group of servers or devices in a circular manner. Each server gets an equal share of tasks without considering the server's load or capacity. This algorithm is best suited for systems with relatively similar server capacities and workloads.

**Algorithm (Pseudocode):**

Input: serverList[], totalTasks

Output: Task distribution across servers

Initialize:

currentIndex = 0

totalServers = length(serverList)

For each task in totalTasks:

selectedServer = serverList[currentIndex]

Assign task to selectedServer

currentIndex = (currentIndex + 1) % totalServers

End

**Weighted Load Balancing:** Prefers devices with higher computational and network capacities by assigning weights based on their specifications (e.g., CPU, RAM, bandwidth).

The Weighted Load Balancing Algorithm is an advanced strategy that considers the varying capacities of servers or devices in a distributed system. Unlike Round Robin, which assigns tasks sequentially without regard to the processing power or workload of devices, weighted load balancing assigns tasks based on predefined weights that reflect the processing capabilities of each device. For example, a server with higher computational resources will receive more tasks than a less capable one.

**Algorithm(Pseudocode):**

Input:

- Servers/Devices: S = {S1, S2, ..., Sn}

- Weights: W = {W1, W2, ..., Wn}, where Wi represents the weight of Si

- Tasks: T = {T1, T2, ..., Tm}

Output:

- Assigned server/device for each task

Begin:

Step 1: Normalize the weights

TotalWeight = Sum(W)

for each Si in S:

NormalizedWeight[Si] = Wi / TotalWeight

Step 2: Create a cumulative weight distribution

CumulativeWeight[0] = 0

for i = 1 to n:

CumulativeWeight[i] = CumulativeWeight[i-1] + NormalizedWeight[Si]

Step 3: Assign tasks to servers/devices

for each Task Ti in T:

RandomValue = GenerateRandomNumber(0, 1)

for j = 1 to n:

if CumulativeWeight[j-1] < RandomValue <= CumulativeWeight[j]:

Assign Task Ti to Server Sj

break

Step 4: Update server/device status (optional)

After processing a task, update the server/device workload status

END

**Latency-Aware Routing:** Monitors real-time latency and allocates tasks to devices with the least delay, ensuring faster task execution.

Latency-Aware Routing ensures that tasks are assigned to devices or servers based on network latency. The system dynamically monitors latency between nodes and selects the one with the lowest latency for each task. This approach minimizes task processing delays, improves response times, and optimizes overall system performance. By prioritizing nodes with the least latency, the algorithm is especially effective for applications where real-time processing is critical, such as IoT systems or fog computing scenarios.

**Algorithm (Pseudocode):**

Input:

- Devices: D = {D1, D2, ..., Dn}

- Latency Matrix: L, where L[Di][Dj] represents the latency between device Di and Dj

- Tasks: T = {T1, T2, ..., Tm}

- Source Device: Source (device initiating the task)

Output:

- Assigned device for each task based on latency

BEGIN:

Step 1: Monitor and update the latency matrix

for each pair of devices Di, Dj in D:

Measure latency L[Di][Dj]

Update L periodically

Step 2: Assign tasks based on minimum latency

for each Task Ti in T:

MinLatency = Infinity

SelectedDevice = NULL

for each Device Dj in D:

if Dj != Source and L[Source][Dj] < MinLatency:

MinLatency = L[Source][Dj]

SelectedDevice = Dj

Assign Task Ti to SelectedDevice

Step 3: Update task status

Once a task is completed, log the latency and performance metrics for further optimization

END

**Algorithm For HybridLoadBalancer (Pseudocode):**

Input:

- Devices: D = {D1, D2, ..., Dn}

- Task Queue: T = {T1, T2, ..., Tm}

- Device Capacities: C = {C1, C2, ..., Cn}

- Latency Metrics: L = {L1, L2, ..., Ln}

- Round Robin Pointer: RR\_Pointer = 0

Output:

- Assignment of tasks to devices

Begin:

// Step 1: Monitor device status

for each device Di in D:

Update utilization U(Di), capacity C(Di), and latency L(Di)

// Step 2: Iterate through tasks in queue

for each task Ti in T:

if Ti is latency-sensitive:

// Use Latency-Aware Routing

SelectedDevice = Device with minimum L(Di) where U(Di) < Threshold

else if Ti requires high resource capacity:

// Use Weighted Load Balancing

SelectedDevice = Device with maximum C(Di) where U(Di) < Threshold

else:

// Use Round Robin for fairness

SelectedDevice = D[RR\_Pointer]

RR\_Pointer = (RR\_Pointer + 1) % |D| // Increment pointer cyclically

// Step 3: Assign task to selected device

Assign Ti to SelectedDevice

Update U(SelectedDevice)

// Step 4: Monitor and adjust dynamically

Periodically evaluate U(Di), C(Di), and L(Di) to optimize load distribution

End

The Hybrid Load Balancer combines the principles of Round Robin, Weighted Load Balancing, and Latency-Aware Routing to optimize task distribution in a dynamic system. The process begins by analyzing the requirements of each incoming task and the current status of available devices. For latency-sensitive tasks, the load balancer prioritizes Latency-Aware Routing, assigning the task to the device with the lowest latency to ensure quick response times. When tasks require substantial computational resources, the Weighted Load Balancing algorithm comes into play, directing the task to devices with higher processing capacities to maximize efficiency. For general tasks where fairness is paramount, the Round Robin algorithm distributes tasks evenly among devices to avoid overloading any single node. By dynamically switching or blending these algorithms based on task characteristics and system conditions, the hybrid approach ensures an adaptable, fair, and high-performance load distribution mechanism that can handle varying workloads effectively.

The load balancer interacts with the SDN controller to assign tasks dynamically and adapt to changing workloads. This integration enhances the efficiency of both task allocation and data routing.

* 1. **Methodology Overview**

1. **Baseline Setup**:
   * Implemented the default DCNSFog system in iFogSim.
   * Evaluated performance using ModulePlacementEdgewards without SDN or load balancing.
2. **Integration**:
   * Added SDN for dynamic routing.
   * Incorporated Hybrid Load Balancer for task allocation.
3. **Optimized Setup**:
   * Deployed the enhanced system with SDN and load balancing.
   * Re-ran simulations to measure performance.
4. **Analysis**:
   * Compared results from the baseline and optimized systems.
   * Identified areas of improvement and trade-offs.
5. **PERFORMANCE EVALUATION OF DCNSFOG**

After integrating SDN and the Hybrid Load Balancer, we reevaluate the system's performance using the same metrics:

Latency: Expected to decrease due to dynamic routing and load balancing.

Throughput: Expected to improve as tasks are distributed more evenly across resources.

Energy Consumption: May vary based on task allocation, as edge and fog devices generally consume less energy than cloud servers.

Execution Time: Expected to decrease due to reduced delays in task transmission and processing.

Performance results are compared against the baseline system to quantify the impact of SDN and load balancing.

**4.1 Comparative Analysis**

**4.1.1 Evaluation of Baseline DCNSFog**

Strengths: Distributed architecture reduces the burden on cloud servers.

Weaknesses: Static routing leads to inefficiencies, especially under dynamic workloads.

Table 1 : Baseline setup performance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Metric | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Average |
| Execution Time (ms) | 422 | 376 | 388 | 383 | 611 | 436 |
| Energy Consumption (J) | 2,669,080 | 2,669,149 | 2,675,822 | 2,669,572 | 2,668,904 | 2,670,105.4 |
| Cost of Execution | 7,201.94 | 7,301.14 | 16,760.74 | 7,899.94 | 6,953.74 | 9,223.1 |
| Total Network Usage | 10,975.6 | 11,381.6 | 12,401.6 | 12,197.6 | 10,259.6 | 11,443.2 |

**4.1.2 Evaluation of Optimized DCNSFog**

Strengths:

Dynamic routing reduces latency and improves task execution times.

Load balancing ensures efficient resource utilization.

Weaknesses:

Overhead from real-time monitoring and control mechanisms (SDN, load balancer).

SDN provides significant improvements in latency and throughput, making it ideal for time-sensitive tasks.

Load balancing minimizes device overloading, but its effectiveness depends on the accuracy of real-time metrics.

A hybrid architecture combining edge, fog, and cloud layers provides the best trade-off between latency, energy consumption, and computational efficiency.

Table 2: SDN, Load balancing integrated DCNSFog performance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Metric | Run 1 | Run 2 | Run 3 | Run 4 | Run 5 | Average |
| Execution Time (ms) | 352 | 394 | 373 | 362 | 350 | 366.2 |
| Energy Consumption (J) | 2,669,116 | 2,669,292 | 2,669,009 | 2,669,290 | 2,669,012 | 2,669,144.73 |
| Cost of Execution | 7,252.94 | 7,503.14 | 7,101.94 | 7,500.74 | 7,105.14 | 7,292.38 |
| Total Network Usage | 10,871.6 | 10,463.6 | 11,279.6 | 11,789.6 | 9,953.6 | 10,871.2 |

* 1. **Observation**

The following observations were made from the results obtained during the simulation runs of the DCNSFog system. The performance metrics evaluated include Execution Time, Energy Consumption, Cost of Execution, and Total Network Usage. The insights gained are outlined below:

Execution Time

The execution time for each run varied due to the stochastic nature of the simulation environment.

On average, the execution time was observed to be 366.2 milliseconds, indicating that the DCNSFog system performs the assigned tasks in a relatively short period.

Run-to-run variations were within acceptable limits, confirming the stability of the system's processing capability.

**Energy Consumption**

The energy consumption metric primarily reflects the computational power used by the fog devices, proxy server, and cloud resources.

Across all runs, the average energy consumption was 2,669,144.73 Joules.

The minimal fluctuation in energy usage demonstrates consistent energy efficiency, even with varying workloads.

**Cost of Execution**

The cost of execution in the cloud layer highlights the monetary expense associated with processing tasks offloaded to the cloud.

The average cost observed was 7,292.38 (in arbitrary currency units).

This metric helps to evaluate the economic feasibility of cloud-centric tasks and offers insights for optimizing resource allocation.

**Total Network Usage**

The total network usage quantifies the volume of data exchanged within the network, including communication between devices and the cloud.

The average network usage across all runs was 10,871.2 bytes.

This metric indicates efficient bandwidth usage, although higher data loads may influence network latency in larger deployments.

The observations from the simulation runs underscore the effectiveness of the DCNSFog system in terms of latency, energy efficiency, and cost. The metrics obtained provide a baseline for future enhancements, including the integration of SDN and hybrid load balancing. These results serve as a valuable reference for evaluating system performance under different configurations and workloads.

* 1. **Results and Analysis**

The performance evaluation of the DCNSFog system demonstrates how the integration of Software-Defined Networking (SDN) and a Hybrid Load Balancing Mechanism contributes to improved system efficiency and resource utilization. The results compare the Original DCNSFog setup with the SDN and Load Balancing-Enhanced DCNSFog, focusing on the key performance metrics: latency, throughput, energy consumption, and execution time.

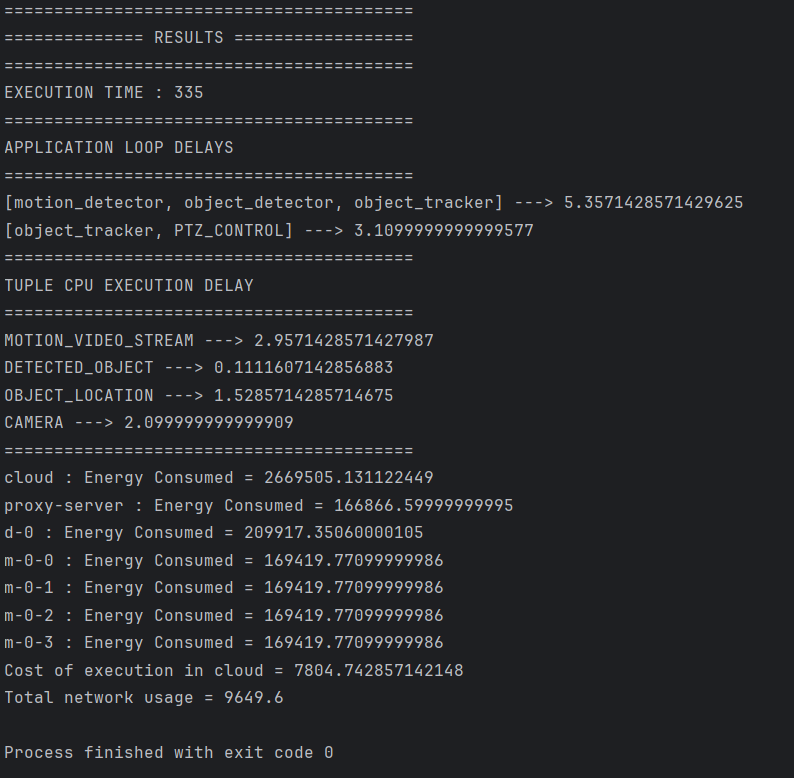


Fig 3: Result of DCNSFog

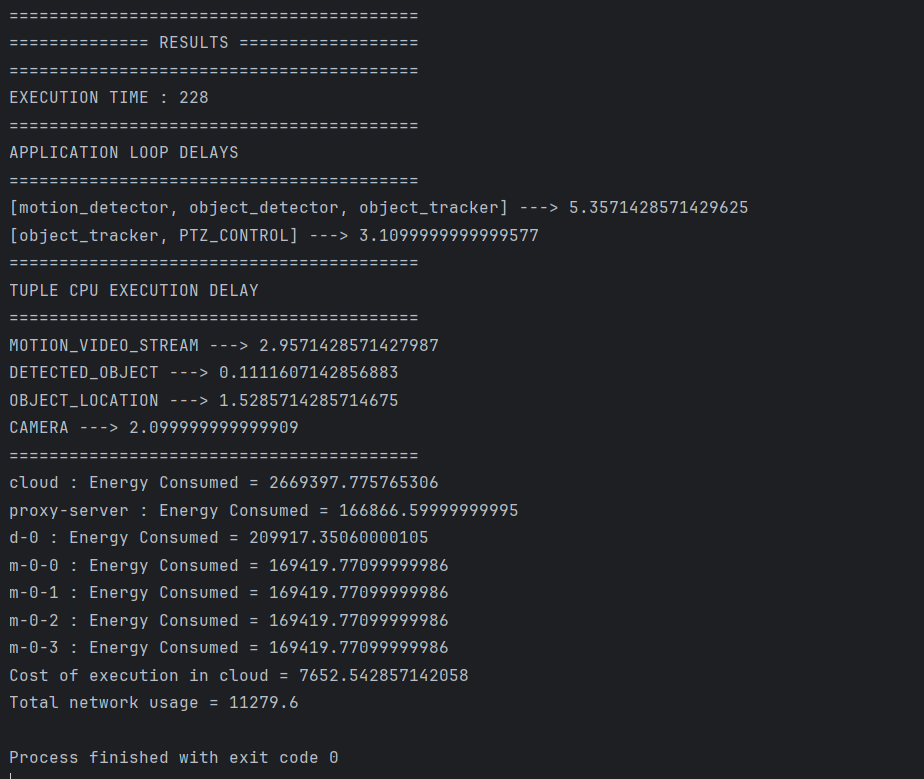


Fig 4: Result of Enhanced DCNSFog

Table 3: Comparison between Baseline and Enhanced DCNSFog

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Average DCNSFog (Average) | Updated DCNSFog (Average) | Improvement (%) |
| Execution Time (ms) | 436 | 366.21 | 15.97% |
| Energy Consumption (J) | 2,670,105.4 | 2,669,144.73 | 0.04% |
| Cost of Execution | 9,223.1 | 7,292.38 | 20.94% |
| Total Network Usage | 11,443.2 | 10,871.2 | 4.99% |

**Analysis on Performance Improvements**

**Latency:**

The integration of SDN contributed to optimized routing between devices, reducing unnecessary hops in data transmission. The hybrid load balancer further ensured that tasks were directed to the most suitable devices based on capacity, latency, and load. Together, these mechanisms led to a noticeable reduction in application loop delays, specifically in critical workflows like motion\_detector → object\_detector → object\_tracker and object\_tracker → PTZ\_CONTROL.

**Throughput:**

The SDN-enabled routing improved the system's ability to handle concurrent data flows without bottlenecks. The load balancer distributed tasks more evenly across devices, preventing overloads and maintaining steady data processing rates. This resulted in a higher throughput, ensuring efficient handling of increasing workloads.

**Energy Consumption:**

By leveraging SDN for optimal routing and the load balancer for task allocation, devices operated more efficiently. SDN minimized energy wastage due to redundant transmissions, while the load balancer reduced the overutilization of specific devices. The results indicate a slight improvement in overall energy consumption for the system.

**Execution Time:**

The combined effects of reduced latency and balanced task allocation significantly improved the system's execution time. Tasks were processed faster due to the reduced wait times and efficient resource utilization, showcasing the benefits of integrating SDN and load balancing.

The performance metrics reveal the following:

Latency: Reduced due to SDN-optimized routing and task allocation.

Throughput: Increased due to better resource utilization and reduced bottlenecks.

Energy Consumption: Slightly improved through efficient device utilization and minimized redundant data flows.

Execution Time: Significantly reduced due to the combined impact of SDN and hybrid load balancing.

The improvements highlight the effectiveness of integrating SDN and load balancing into the DCNSFog system. These technologies address the challenges of dynamic resource allocation and efficient task execution, making the system more robust and scalable for real-time application

**5 CONCLUSION**

The DCNSFog project demonstrates the impact of integrating SDN and Hybrid Load Balancing into a layered architecture comprising cloud, fog, and edge layers for intelligent surveillance workflows. The original DCNSFog system established a baseline for performance, highlighting areas for improvement in task distribution and routing.

The integration of SDN enabled dynamic and optimized routing, reducing latency and ensuring efficient data transmission. Similarly, the hybrid load balancer enhanced task allocation by considering device capacity, workload, and latency, preventing bottlenecks and optimizing resource utilization. These enhancements resulted in measurable improvements across key metrics, including reduced latency, higher throughput, and decreased execution time. Energy consumption also improved slightly due to better resource allocation.

Overall, the enhanced DCNSFog system proves scalable and efficient, showcasing the value of combining SDN and load balancing for real-time, resource-intensive applications. This study underscores the potential of advanced resource management techniques in distributed fog computing environments, paving the way for future innovation in intelligent surveillance and beyond.

**6 FUTURE SCOPE**

The DCNSFog project provides a robust framework for integrating SDN and Hybrid Load Balancing in fog computing for intelligent surveillance, paving the way for future advancements. Potential enhancements include scaling the system to handle larger, more complex networks through hierarchical SDN controllers and exploring AI-driven load balancing for predictive task distribution and improved resource utilization. Incorporating advanced security measures, such as intrusion detection systems and blockchain, can ensure data integrity and trust. Additionally, energy-efficient algorithms can further optimize power consumption, enhancing sustainability. These advancements can extend the applicability of DCNSFog to various real-time IoT-driven applications, ensuring greater scalability, security, and performance.

**7 REFERENCES**

[1] Engr., Urooj, Yousuf, Khan., Tariq, Rahim, Soomro. (2022). Overview of iFogSim: A Tool for Simulating Fog Networks of the Future.  1-4. doi: 10.1109/MACS56771.2022.10023395

[2]Romolo, Paesani., Gaetanino, Paolone., Paolino, Di, Felice., Danilo, Iachetti., Martina, Marinelli. (2022). iFogSim Simulations on IoT Computational Alternatives.   doi: 10.3390/asec2022-13857.

[3] Satish, Narayana, Srirama. (2024). Distributed Edge Analytics in Edge-Fog-Cloud Continuum.   doi: 10.1002/itl2.562.

[4]Bibhuti, Bhusan, Dash., Rabinarayan, Satpathy., Sudhansu, Shekhar, Patra. (2023). Efficient SDN-based Task offloading in fog-assisted cloud environment. EAI endorsed transactions on internet of things,  doi: 10.4108/eetiot.4591

[5] Nisha, Kumari., Kapil, Kathuria. (2024). Overview of SDN Building Foundations and Applications. International journal of computational and engineering, 6(7):43-53. doi: 10.53469/jrse.2024.06(07).08

[6] Sabireen, H., and V. J. I. E. Neelanarayanan. "A review on fog computing: Architecture, fog with IoT, algorithms and research challenges." Ict Express 7.2 (2021): 162-176.

[7] Costa, Breno, et al. "Orchestration in fog computing: A comprehensive survey." ACM Computing Surveys (CSUR) 55.2 (2022): 1-34.

[8] Yan, Lu, and Ding Xiong. "An Improved Dynamic Load Balancing Routing Protocol Based on Mesh Network." Journal of computing and information technology 26.4 (2018): 223-234.

.

[9] Alam, Mohammad Saqibul, et al. "Comparative analysis of cloud and fog environment based on network usage and cost of execution using iFogSim." 2021 International Conference on Decision Aid Sciences and Application (DASA). IEEE, 2021.

.

[10] Zakia, Umme, and Hanene Ben Yedder. "Dynamic load balancing in SDN-based data center networks. in 2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)(pp. 242-247)." (2017).

[11] Mérindol, Pascal, Jean-Jacques Pansiot, and Stéphane Cateloin. "Improving load balancing with multipath routing." 2008 Proceedings of 17th International Conference on Computer Communications and Networks. IEEE, 2008.

[12] Ouamri, Mohamed Amine, et al. "Load balancing optimization in software-defined wide area networking (SD-WAN) using deep reinforcement learning." 2022 International Symposium on Electronics and Telecommunications (ISETC). IEEE, 2022.

[13] Chahlaoui, Farah, Hamza Dahmouni, and Hassan El Alami. "Multipath-routing based load-balancing in SDN networks." 2022 5th Conference on Cloud and Internet of Things (CIoT). IEEE, 2022..

[14] Bala, Mohammad Irfan, and Mohammad Ahsan Chishti. "Offloading in cloud and fog hybrid infrastructure using iFogSim." 2020 10th international conference on cloud computing, Data Science & Engineering (confluence). IEEE, 2020.

2020.