Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Delay-Aware Dynamic Resource Orchestration for IoT-Enabled Software-Defined Edge Networks

Lalita Agrawal[1] | **Ayan Mondal**[1] | **Mohammad S. Obaidat, Life Fellow of IEEE**[2] | **Erkki Harjula**[3]

[1]Department of Computer Science and Engineering, Indian Institute of Technology Indore, Madhya Pradesh, India

[2]Distinguished Professor, The King Abdullah II School of Information Technology, The University of Jordan, Amman 11942, Jordan and School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China, Department of Computational Intelligence, School of Computing, SRM University, SRM Nagar, Kattankulathur 603203, TN, India, and School of Engineering, The Amity University, Noida, UP 201301, India. Email: m.s.obaidat@ieee.org

[3]Centre for Wireless Communications, University of Oulu, Finland

Correspondence
Corresponding author: Lalita Agrawal.
Email: phd2101201003@iiti.ac.in

Abstract

In the rapidly evolving Internet of Things (IoT) ecosystem, the integration of Software-Defined Networking (SDN) with edge computing is critical for optimizing performance in IoT applications. This paper introduces a novel framework, named D-RESIN, designed to dynamically orchestrate resources within IoT-enabled SDN at the edge, explicitly focusing on minimizing delays. The proposed framework employs evolutionary game theory to manage and optimize resource allocation across IoT devices, Open vSwitches, and Edge nodes. We implemented the proposed D-RESIN schemes using the Mininet network emulator with Ryu SDN controller and Open vSwitches. We found out that D-RESIN reduces average processing delay at the access tier by 52.43-88.82% and 32.71-87.91% compared to the existing scheme — T-RESIN and FlowMan, respectively. At the edge tier, D-RESIN decreases the average processing delay by 35.44-85.10% compared to T-RESIN. These simulation results highlight the effectiveness of D-RESIN in enhancing scalability and efficiency for delay-sensitive IoT applications.

KEYWORDS
Internet of Things (IoT), Software-Defined Networking (SDN), Edge Computing, Evolutionary Game Theory, Delay-Sensitive Applications.

1 | INTRODUCTION

The Internet of Things (IoT) represents a transformative evolution in the digital landscape, where interconnected devices and sensors seamlessly integrate to collect, exchange, and process data in real-time [1,2]. Traditional IoT architectures relying on centralized cloud computing suffer from high latency and response time issues, making them unsuitable for delay-sensitive applications [3,4,5,6]. These limitations hinder real-time processing for IoT applications at the edge. Hence, we plan to integrate IoT with Software-Defined Networking (SDN) at the edge computing environment that decouples the network control plane from the data plane [7,8,9,10]. As per the current scenario, traditional IoT network infrastructures often struggle with latency issues and cannot meet the dynamic resource demands of IoT devices effectively [11,12]. In this work, we plan to enhance the scalability and efficiency of edge computing environments by adjusting the resource allocation of the network based on real-time data traffic and application demands. We design a framework for IoT-enabled Software-Defined Edge Networks (SDENs) that ensures optimal performance to address the challenges of delay-aware resource orchestration.

For SDEN networks, the SDN controller[‡] has two types of application programming interfaces (APIs) for three-tier architecture— northbound and southbound APIs. We consider the OpenFlow protocol to be a southbound interface between Open vSwitches and the Ryu SDN controller. We consider REST API to be a northbound interface between the Ryu SDN

Abbreviations: SDEN, Software-Defined Edge Networks; IoT, Internet of Things; SDN, Software-Defined Networks.
[‡] https://opennetworking.org/software-defined-standards/

controller and applications. Open-source Apache 2.0[§] has licensed Open vSwitches as a high-quality multilayered switch for network orchestration. The Ryu controller is also actively developed for research activities like SDN OpenFlow rules placement management in its ternary content addressable memory (TCAM) compared to other SDN controllers like Pox.

Motivating Scenario: An extensive network of IoT devices is deployed to observe traffic flow, aiming to optimize channel delay and alleviate congestion [13,14]. For example, during a major public event such as the Olympics, a vast array of 6G-enabled IoT sensors and cameras can be set up throughout the city to provide real-time data on traffic conditions. This paper presents a novel approach where edge networks dynamically allocate resources in real-time, significantly reducing latency. Our method effectively manages data surges during large-scale events or emergencies, ensuring uninterrupted connectivity and efficient traffic management. The rapid response capabilities of 6G technology allow for seamless adjustments to traffic signals and rerouting, preventing bottlenecks and enhancing overall traffic flow.

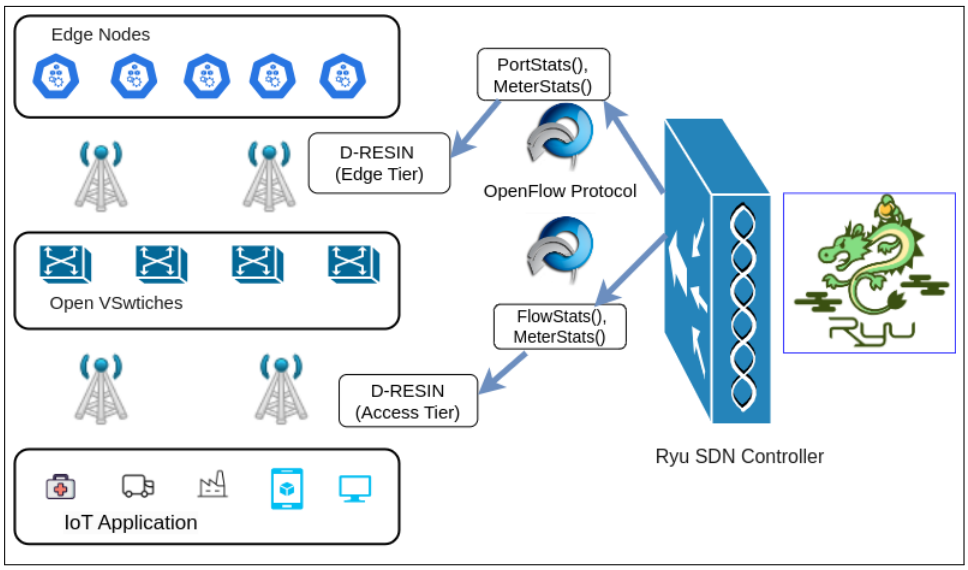

FIGURE 1 Schematic architecture of IoT-enable SD-Edge networks

In this paper, we introduce Delay-Aware Dynamic Resource Orchestration for IoT-Enabled SDEN, named D-RESIN, a novel framework to orchestrate resources in SDENs, focusing on minimizing computation delays. As shown in Figure 1, functions FlowStats() and PortStats() are used for the statistics collection of data at the Access tier and Edge tier, respectively. Similarly, MeterStats() is used for bandwidth establishment between IoT devices to Open vSwitches and Open vSwitches to Edge nodes. We execute the D-RESIN algorithms at access and edge tiers to dynamically adjust the network's resource allocation based on real-time data traffic and application demands. D-RESIN addresses the latency issues inherent in IoT deployments and significantly enhances the scalability and efficiency of edge computing environments. The contributions of this article are as follows:

1. We present D-RESIN, a dynamic resource allocation scheme for delay-sensitive IoT applications in the SDEN network.
2. We use an evolutionary game theoretic approach for resource allocation in IoT-enabled SDEN. The proposed scheme ensures optimal processing delay between IoT devices, Open vSwitches, and Edge nodes— are programmed by the Ryu SDN controller.
3. We evaluate the existence of evolutionary equilibrium for D-RESIN in access and edge tiers.
4. We implemented our proposed schemes using the Mininet emulation environment. Our approach significantly reduces delay compared to existing schemes and achieves superior quality of service (QoS) for delay-sensitive applications.

[§] https://www.apache.org/

2 | RELATED WORK

2.1 | Consistency and Delay challenges in SDN

Software-defined network (SDN) maintains the consistency for distributed control plane architecture as features are no longer limited to what the vendor provides. In this section, we explore strategies and solutions proposed in recent literature to address the inherent trade-offs between real-time network performance and the consistency required for reliable SDN operations. For the purpose mentioned above, Mondal *et al.* [15] designed a FlowMan scheme for managing data flows in the presence of elephant and mice flows (Heterogeneous environment). Using the bisection method, the authors used the bounded knapsack approach to associate data flows from IoT devices to SDN switches and ensured optimal network delay and throughput. Another work by Zhang and Zhu [16] proposed an SDN-based framework integrating network function virtualization (NFV) with WiFi and device-to-device (D2D) offloading to enhance statistical QoS for multimedia services in 5G. Balasubramanian *et al.* [14] presented an SDN-based architecture, named TSN_u, aimed at enhancing the performance of time-sensitive industrial IoT networks. The framework is grounded in the IEEE Time Sensitive Networking (TSN) standards, particularly leveraging the IEEE 802.1Qbv and IEEE 802.1Qcc standards for improving transmission time-slot allocations, congestion mitigation, and network stability. Mondal *et al.* [17] proposed an analytical model named OPUS, alongside a specific queuing scheme, to theoretically determine the minimal buffer size requirements of an OpenFlow switch. Extensive simulations demonstrated that an optimized buffer size can significantly enhance packet handling rates while maintaining acceptable packet waiting times. Saha *et al.* [18] proposed a traffic-aware Quality of Service (QoS) routing scheme within a Software-Defined IoT network framework. It specifically addressed delay-sensitive and loss-sensitive routing strategies for incoming packets from IoT devices.

TABLE 1 Comparison of Existing Delay Sensitive Scheme for IoT Network

Existing Work for Delay Sensitive Scheme	Software-Defined Edge Network Features				
	Consistency	Dynamic Configuration	Edge Computing	Three Tiers	Delay
Balasubramanian *et al.* [14]	✓	✗	✗	✗	✓
Mondal *et al.* [15]	✓	✓	✗	✗	✓
Mondal *et al.* [17]	✓	✓	✗	✗	✓
Xavier *et al.* [19]	✗	✓	✓	✗	✓
Jain *et al.* [20]	✗	✓	✓	✓	✗

2.2 | Dynamic and Three-Tier Architecture for Edge Network

The edge computing environment provides a decentralized approach to resource allocation and data processing at the network's periphery/edge. Here, we review recent advancements in dynamic resource allocation methods and integrate a three-tier architecture—consisting of cloud, edge, and end devices—to optimize data processing and latency for SDN-enabling technology. For the same, Xavier *et al.* [19] addressed the challenge of managing computing resources in cloud, edge, and IoT environments to meet the latency and energy efficiency demands of time-sensitive applications. The proposed algorithm is evaluated against non-collaborative and collaborative offloading approaches for edge node utilization. Agrawal *et al.* [21] proposed T-RESIN, an evolutionary game-based scheme that optimizes quality of service and throughput by managing resources at both local and edge tiers effectively in the SDEN network. However, it acknowledged limitations in handling delay-sensitive tasks. In another work, Jain *et al.* [20] presented a metaheuristic approach with a blockchain-based resource allocation technique (MWBA-RAT) for Cybertwin-driven 6G on the Internet of Everything (IoE) environment, addressing the challenges posed by increasing mobile Internet traffic and service demands. The proposed technique utilizes a quasi-oppositional search and rescue optimization (QO-SRO) algorithm for enhancing the network's ability to effectively monitor, manage, and share limited spectrum resources. Liyanage *et al.* [22] explored the integration of MEC in 5G networks to enhance IoT applications. The study addressed the challenges and future directions for realizing a cohesive MEC-IoT ecosystem in 5G infrastructures. Manogaran *et al.* [23] presented a novel resource allocation method with optimal fog node placement in IoT-Fog-Cloud architecture, aiming to minimize service

TABLE 2 List of Symbols

Symbol	Description
$\mathcal{N}$	Set of IoT devices/things
$\mathcal{S}$	Set of SDN switches
$\mathcal{E}$	Set of Edge nodes
B_s	Bandwidth associated with each switch $s \in \mathcal{S}$
F_n	Set of flows generated by each IoT device $n \in \mathcal{N}$
R_s^{max}	Maximum number of flow rules in TCAM of switch $s \in \mathcal{S}$
$V(s)$	Volume of data associated with each switch $s \in \mathcal{S}$
$Pd_s^{avg}(f, n, s)$	Average processing delay of each SDN switch $s \in \mathcal{S}$ for any flow f
μ_s	Processing rate for all SDN switch
$V(e)$	Volume of data associated with each edge node $e \in \mathcal{E}$
$C_{use}(e)$	Computational power used by an edge node $e \in \mathcal{E}$
$Pd_e^{avg}(f, s, e)$	Average processing delay of each edge node $e \in \mathcal{E}$ for any flow f
μ_e	Processing rate for every edge node
E_{use}^e	Total energy consumption of each edge node $e \in \mathcal{E}$
E_{res}^e	Residual energy of each edge node $e \in \mathcal{E}$
$x_s(\cdot)$	Population share of each switch $s \in \mathcal{S}$
$U_s(\cdot)$	Utility function for each switch $s \in \mathcal{S}$
$\dot{x_s}(\cdot)$	Replicator dynamics for SDN Switches
$y_e(\cdot)$	Population share of each edge node $e \in \mathcal{E}$
$W_e(\cdot)$	Utility function for each edge node $e \in \mathcal{E}$
$\dot{y_e}(\cdot)$	Replicator dynamics for edge nodes
α, β	Evolutionary control factor

delays and resource exploitation. Sami *et al.*[24] introduced IScaler, a deep reinforcement learning-based solution for intelligent and proactive resource scaling and service placement in Mobile Edge Computing (MEC) environments.

2.3 | Synthesis

In the existing literature, a significant number of researchers have introduced flow association schemes tailored for delay-sensitive applications within SDN frameworks. However, these methodologies proved impractical for dynamic resource orchestration in edge computing contexts. We present a comparative analysis of the existing delay-sensitive scheme based on SDEN features in Table 1. Hence, we present a three-tier architecture—comprising IoT devices, SDN switches, and Edge Nodes in this work that incorporate all the SDEN features.

3 | SYSTEM MODEL

We consider an IoT-enabled SD-edge network infrastructure. It has only one SDN controller in the control plane and multiple switches in the data plane. The computation of data generated from IoT devices/things occurs at edge nodes. Data packets/flows from IoT devices to edge nodes go through switches via access points. We consider $\mathcal{N}$, $\mathcal{S}$, and $\mathcal{E}$ as a set of IoT devices, SDN switches, and edge nodes, respectively. Bandwidth associated with each switch $s \in \mathcal{S}$ is represented as B_s such that total bandwidth $\mathcal{B}$ distributed among all the switches is as follows:

$$\mathcal{B} = \sum_{s \in \mathcal{S}} B_s \tag{1}$$

Each IoT device $n \in \mathcal{N}$ generates F_n set of flows. Each flow $f_i^n \in F_n$ starts and ends its transmission from the IoT device $n \in \mathcal{N}$ at $t_s(f_i^n)$ and $t_e(f_i^n)$, respectively, where $i \in (\mathbb{Z}^+ \cap [0, F_n])$ and $t_e(f_i^n) > t_s(f_i^n)$. Flows are dynamic and can be processed at any SDN switch. R_s^{max} denotes the maximum number of flow rules that switch $s \in \mathcal{S}$ can install in its ternary content addressable memory (TCAM). F_s represents the number of flow rules associated with each switch $s \in \mathcal{S}$ that satisfies the constraint $F_s \leq R_s^{max}$.

The processing delay[¶] of any flow depends on queue discipline and volume of the data associated with a particular flow. For queue discipline, we consider first-come-first-serve, where flows from the queue are selected for servicing based on their arrival. The processing rate μ_s represents how much volume of data can be processed per unit of time at switch s, and it is constant and does not depend on network parameters. The volume of data associated with each SDN switch $s \in \mathcal{S}$ is represented in Equation (2). Definition 1 shows switch processing delay for active flow rules F_s, which eventually helps in determining the average processing delay of associated flows represented by Equation (4).

$$V(s) = \sum_{f \in \bigcup_n F_n} a_{f,n,s} V_f \leq B_s \tag{2}$$

where $a_{f,n,s}$ is defined as follows:

$$a_{f,n,s} = \begin{cases} 1, \text{ if flow } f \text{ of } n \in \mathcal{N} \text{ associated to a switch } s \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

Definition 1. We define *stat.duration_sec* parameter of AggregateStats() function of the Open vSwitches $s \in \mathcal{S}$ to capture the overall processing delay for F_s set of flow rules that stay active at any time instant t in its TCAM memory.

$$Pd_s^{avg}(f, n, s) = \frac{V(s)}{\mu_s \sum\limits_{f \in \bigcup_n F_n} a_{f,n,s}} \tag{4}$$

We consider that flow $f_j^s \in F_s$ is transmitted from the associated SDN switch towards the edge node for resource allocation when it completes its processing at the switch $s \in \mathcal{S}$ where $j \in (\mathbb{Z}^+ \cap [0, F_s])$. Each edge node $e \in \mathcal{E}$ has resource constraints in the form of computation C_e and memory M_e for processing the flows received from IoT devices. The volume of data and computational power used by an edge node $e \in \mathcal{E}$ for processing the flows are represented in Equations (5) and (6).

$$V(e) = \sum_{f \in \bigcup_s F_s} b_{f,s,e} V_f \leq M_e \tag{5}$$

$$C_{use}(e) = \sum_{f \in \bigcup_s F_s} b_{f,s,e} C_f \leq C_e \tag{6}$$

where $b_{f,s,e}$ is defined as follows:

$$b_{f,s,e} = \begin{cases} 1, \text{ if flow } f \text{ of } s \in \mathcal{S} \text{ associated to edge node } e \\ 0, \text{ otherwise} \end{cases} \tag{7}$$

Definition 2. We define *stat.duration_sec* parameter of PortStats() function to capture the overall processing delay for F_e set of flow rules that are associated with the edge nodes $e \in \mathcal{E}$ for resource allocation (memory and computation).

The average processing delay of an edge node $e \in \mathcal{E}$ is presented in Equation (8) where μ_e is the processing rate for every edge node.

$$Pd_e^{avg}(f, s, e) = \frac{V(e)}{\mu_e C_{use}(e) \sum\limits_{f \in \bigcup_s F_s} b_{f,s,e}} \tag{8}$$

Total energy consumption E_{use}^e for computing and storing any flow associated with an edge node e is represented in Equation (9). At a particular time instant t, residual energy E_{res}^e of edge node e is written as follows:

$$E_{use}^e = \sum_{f \in \bigcup_s F_s} b_{f,s,e} E(f, s, e) \leq E^e \tag{9}$$

[¶] We have not considered channel delay (transmission and propagation delay).

$$E^e_{res} = E^e - \sum_{f \in \bigcup_s F_s} b_{f,s,e} E(f,s,e) \tag{10}$$

Here, E^e denotes the total energy of each node e. Each edge node is constrained by its residual energy for the association of incoming flows. Table 2 summarizes the main symbols and their descriptions, which are frequently used throughout the paper.

4 | D-RESIN: THE PROPOSED DELAY-AWARE RESOURCE ORCHESTRATION SCHEME

We use the *evolutionary game theoretic approach*[25] to model the interaction among IoT devices and SDN switches and SDN switches to edge nodes. D-RESIN continuously monitors network conditions and adjusts data flow between IoT devices, Open vSwitches, and Edge nodes based on real-time demands. The framework distributes workloads to minimize congestion and ensure faster processing at access and edge tiers.

4.1 | Justification for Using Evolutionary Game

We observe that it requires a binary variable $a_{f,n,s}$ in Equation (4) for minimizing the overall processing delay between IoT devices and SDN switches. Similarly, for SDN switches to the edge node, it requires a binary variable $b_{f,s,e}$ in Equations (8), (9), and (10) for minimizing the delay. So the problem mentioned above is a multiple *binary integer programming problem*[26]. It can be mapped to the variant of the *0-1 knapsack problem*[26,27], a well-known NP-complete problem. Traditional brute-force or conventional optimization approaches are impractical for large-scale IoT deployments due to their exponential complexity. These methods can not efficiently handle the real-time resource allocation challenges posed by fluctuating IoT device demands and network traffic. To address this issue, we use *evolutionary game theoretic approach* to provide a computationally feasible solution with polynomial-time complexity while enabling the system to adapt dynamically according to network conditions. The *evolutionary game* becomes a method for exploring and optimizing the delay of the *0-1 knapsack problem*, which in turn represents the optimized solution for the original binary integer programming problem.

4.2 | Game Formulation

We aim to minimize the processing delay $Pd^{avg}_s(f,n,s)$ of any flow from IoT devices to switches and the processing delay $Pd^{avg}_e(f,s,e)$ of any flow from SDN switches to edge nodes, which is represented in Equations (4) and (8), respectively. We consider each IoT end-device $n \in \mathcal{N}$ to act as a player and select the SDN switch to forward its data flows from the IoT end user to the Edge node with the help of the SDN controller. The population share $x_s(\cdot)$ of each switch $s \in \mathcal{S}$ is represented in Equation (11), which is a function of average processing delay and flows associated with the switch $s \in \mathcal{S}$. Similarly, the population share $y_e(\cdot)$ of edge nodes $e \in \mathcal{E}$ is represented in Equation (12).

$$x_s(\cdot) = \frac{\sum_{f \in F_s, \forall n} a_{f,n,s} Pd^{avg}_s(f,n,s)}{\sum_{f \in \bigcup_s F_s, \forall n} a_{f,n,s} Pd^{avg}_s(f,n,s)} \tag{11}$$

$$y_e(\cdot) = \frac{\sum_{f \in F_e, \forall s} b_{f,s,e} Pd^{avg}_e(f,s,e)}{\sum_{f \in \bigcup_e F_e, \forall s} b_{f,s,e} Pd^{avg}_e(f,s,e)} \tag{12}$$

Utility Function of SDN Switches and Edge Nodes

For delay-sensitive applications, we need to minimize the population shares $x_s(\cdot)$ and $y_e(\cdot)$ of SDN switches and Edge nodes, respectively. Hence, the utility function of SDN switches $U_s(\cdot)$ is considered a negative payoff of SDN switches. This means that switches with lower processing delays and fewer flow rules will have higher utility or payoff values, while switches with higher

processing delays and more flow counts will have lower utility values. We define the utility function $U_s(\cdot)$ for each switch and the average payoff $\overline{U}(\cdot)$ of the population for SDN switches as follows:

$$U_s(\cdot) = x_s(\cdot)\left(1 - \frac{F_s}{R_s^{max}}\right) \tag{13}$$

$$\overline{U}(\cdot) = \sum_{s\in\mathcal{S}} x_s(\cdot)U_s(\cdot) \tag{14}$$

Processing delay and consumed energy of each Edge node define the negative fitness or payoff for the Edge nodes. A lower fitness or payoff value indicates a better outcome for edge nodes. The utility function $W_e(\cdot)$ for each edge node and the average payoff $\overline{W}(\cdot)$ of the population for edge nodes are represented as follows.

$$W_e(\cdot) = y_e(\cdot)\left(1 - \frac{E_{use}^e}{E^e}\right) \tag{15}$$

$$\overline{W}(\cdot) = \sum_{e\in\mathcal{E}} y_e(\cdot)W_e(\cdot) \tag{16}$$

Hence, the objective functions of each switch $s \in \mathcal{S}$ and edge node $e \in \mathcal{E}$ are as follows:

$$\arg_{x_s} \min U_s(\cdot) \tag{17}$$

$$\arg_{y_e} \min W_e(\cdot) \tag{18}$$

Replicator Dynamics

Replicator dynamics is used to model the evolution of strategies in a population of Edge nodes and SDN switches to optimize network performance. D-RESIN dynamically adjusts the population shares of SDN switches and edge nodes via replicator dynamics, which balances the load and minimizes processing delays even under high traffic. Replicator dynamics for SDN switches $\dot{x_s}(\cdot)$ and edge nodes $\dot{y_e}(\cdot)$ are defined as follows:

$$\dot{x_s}(\cdot) = \alpha x_s(\cdot)\big(U_s(\cdot) - \overline{U}(\cdot)\big) \tag{19}$$

$$\dot{y_e}(\cdot) = \beta y_e(\cdot)\big(W_e(\cdot) - \overline{W}(\cdot)\big) \tag{20}$$

where $\alpha > 0$ and $\beta > 0$ are evolutionary control factors.

4.3 | Existence of Evolutionary Equilibrium for D-RESIN Scheme

Theorem 1. *For a given set of flow rules F_s and the average processing delay $Pd_s^{avg}(f, n, s)$ of a switch, there exists an evolutionary equilibrium for SDEN network switches at the Access tier.*

Proof. We aim to minimize $Pd_s^{avg}(f, n, s)$, which ensures optimal data traffic distribution among SDN switches based on processing delay. To identify the existence of evolutionary equilibrium, the replicator dynamics equation for switches should be as follows:

$$\dot{x_s}(\cdot) = \alpha x_s(\cdot)\big(U_s(\cdot) - \overline{U}(\cdot)\big) = 0 \tag{21}$$

Here, the population share of each switch $s \in \mathcal{S}$ is $x_s(\cdot) \geq 0$, and the evolutionary control factor α is a positive constant, i.e. $\alpha > 0$. Therefore, Equation (21) is represented as follows:

$$U_s(\cdot) - \overline{U}(\cdot) = 0 \tag{22}$$

By expanding the utility and average utility function of switches in Equation (22), we get:

$$(x_s)^2 - x_s + \frac{\sum\limits_{s' \in \mathcal{S}/\{s\}} (x_{s'})^2 \left(1 - \frac{F_{s'}}{R_{s'}^{max}}\right)}{\left(1 - \frac{F_s}{R_s^{max}}\right)} = 0 \tag{23}$$

In order to find the evolutionary equilibrium, we first identify the coefficients of the quadratic Equation (23). By solving using a quadratic formula, we yield optimal population share x_s^* for each switch $s \in \mathcal{S}$ as follows:

$$x_s^* = \frac{1 \pm \sqrt{1 - 4\psi}}{2} \tag{24}$$

where $\psi = \left[\frac{\sum\limits_{s' \in \mathcal{S}/\{s\}} (x_{s'})^2 \left(1 - \frac{F_{s'}}{R_{s'}^{max}}\right)}{\left(1 - \frac{F_s}{R_s^{max}}\right)}\right]$. □

Theorem 2. *For a given set of flow rules F_e associated with edge nodes, there exists an evolutionary equilibrium at the edge tier for the average processing delay $Pd_e^{avg}(f, s, e)$.*

Proof. At evolutionary equilibrium, replicator dynamics reach zero at the edge tier. Hence, Equation (20) is written as follows:

$$\dot{y_e}(\cdot) = \beta y_e(\cdot)\big(W_e(\cdot) - \overline{W}(\cdot)\big) = 0 \tag{25}$$

Similarly, for edge nodes, we consider the population share is $y_e(\cdot) \geq 0$, and the evolutionary control factor is $\beta > 0$. Hence, we get:

$$W_e(\cdot) - \overline{W}(\cdot) = 0 \tag{26}$$

By placing the payoff and average utility values of edge nodes in the above Equation (26), we get:

$$(y_e)^2 - y_e + \frac{\sum\limits_{e' \in \mathcal{E}/\{e\}} (y_{e'})^2 \left(1 - \frac{E_{use}^{e'}}{E^{e'}}\right)}{\left(1 - \frac{E_{use}^{e}}{E^{e}}\right)} = 0 \tag{27}$$

At evolutionary equilibrium, we yield optimal population share y_e^* for each edge node $e \in \mathcal{E}$ as follows:

$$y_e^* = \frac{1 \pm \sqrt{1 - 4\kappa}}{2} \tag{28}$$

where $\kappa = \left[\frac{\sum\limits_{e' \in \mathcal{E}/\{e\}} (y_{e'})^2 \left(1 - \frac{E_{use}^{e'}}{E^{e'}}\right)}{\left(1 - \frac{E_{use}^{e}}{E^{e}}\right)}\right]$ □

4.4 | Proposed Algorithms

For our proposed SDEN network architecture, we have designed two D-RESIN algorithms. As shown in the workflow illustrated in Figure 2, the decision point (access tier) ensures — access and edge tiers executing Algorithms 1 and 2, respectively, with the help of the Ryu controller. The Knapsack algorithm identifies the association among IoT devices, SDN switches, and Edge computing nodes. To optimize the processing delay, each switch and edge node need to strategize $x_s(\cdot)$ and $y_e(\cdot)$, respectively, to reach evolutionary equilibrium. We set the evolutionary control factor $\alpha = 0.01$ at the access tier and $\beta = 0.1$ at the edge tier. We terminate the execution of both algorithms when replicator dynamics are $\dot{x_s}(\cdot) \not\approx 0$ and $\dot{y_e}(\cdot) \not\approx 0$ ($\leq 10^{-6}$ practically). At evolutionary equilibrium, we evaluate all the SDEN network parameters.

4.5 | Complexity Analysis

The main part of the Algorithm 1 involves a *do-while* loop, from lines 4-16, which continues until an evolutionary stable state ω is reached. Within the outer loop, line 5 takes constant time $O(1)$. For lines 7-9, each line evaluates an equation that takes

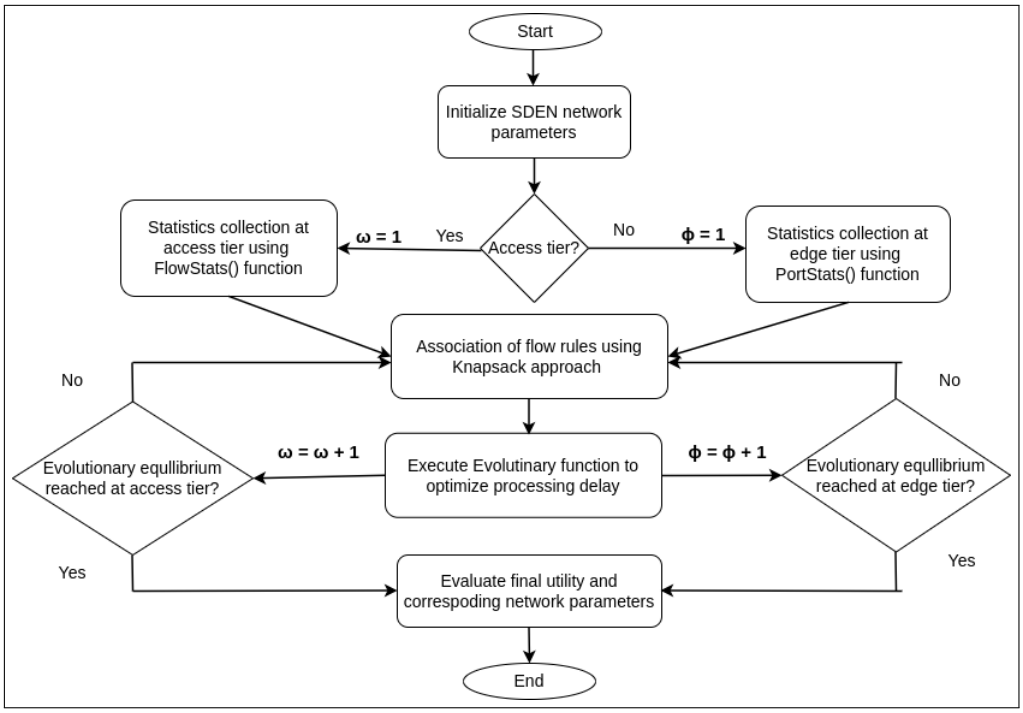

FIGURE 2 Workflow diagram of D-RESIN

Algorithm 1 D-RESIN for Access Tier

INPUTS:

Network and SDN switch parameters: $\mathcal{N}, \mathcal{S}, F_s, \mu_s, V(s), R_s^{max}, \alpha$

OUTPUTS:

Optimize population share and utility value vector for SDN switches: x^*, U^*

PROCEDURE:

$\omega \leftarrow 0$
Distribute flows $0 \leq i \leq f_n$, where $n \in \mathcal{N}$, randomly across switches $s \in \mathcal{S}$.
do
  $\omega \leftarrow \omega + 1$
  for Each $s \in \mathcal{S}$ **do**
    Evaluate $Pd_s^{avg}(f, n, s)$ using Equation (4)
    Determine population share $x_s(\cdot)$ as per Equation (11)
    Evaluate utility value $U_s(\cdot)$ according to Equation (13)
  end for
  Evaluate average utility value $\overline{U}(\cdot)$ using Equation (14)
  for Each $s \in \mathcal{S}$ **do**
    Evaluate replicator dynamic $\dot{x_s}(\cdot)$ using Equation (19)
    Update population share for the next iteration as: $x_s(\cdot) \leftarrow x_s(\cdot) + \dot{x_s}(\cdot)$
  end for
while $(\dot{x_s}(\cdot) \not\approx 0)$
Get evolutionary equilibrium at iteration ω
Finalize population share as: $x_s^* \leftarrow x_s(\cdot)$
Evaluate final utility value U_s^* using Equation (13) at evolutionary iteration ω
return x^*, U^*;

Algorithm 2 D-RESIN for Edge Tier

INPUTS:

1: Network and Edge Nodes parameters: $\mathcal{N}, \mathcal{S}, \mu_e, C_{use}(e), V(e), E^e, \beta$

OUTPUTS:

1: Optimize population share and utility value vector for Edge nodes: y^*, W^*

PROCEDURE:

1: $\phi \leftarrow 0$
2: Distribute flows $0 \leq i \leq f_s$, where $s \in \mathcal{S}$, randomly across edge nodes $e \in \mathcal{E}$ for computational resources.
3: **do** $\phi \leftarrow \phi + 1$
4: **for** Each $e \in \mathcal{E}$ **do**
5: Evaluate $Pd_e^{avg}(f, s, e)$ using Equation (8)
6: Evaluate E_{use}^e using Equation (9)
7: Determine population share $y_e(\cdot)$ using Equation (12)
8: Evaluate utility value $W_e(\cdot)$ using Equation (15)
9: **end for**
10: Evaluate average utility value $\overline{W}(\cdot)$ using Equation (16)
11: **for** Each $e \in \mathcal{E}$ **do**
12: Evaluate replicator dynamic $\dot{y_e}(\cdot)$ using Equation (20)
13: Update population share for the next iteration as follows: $y_e(\cdot) \leftarrow y_e(\cdot) + \dot{y_e}(\cdot)$
14: **end for**
15: **while** $(\dot{y_e}(\cdot) \not\approx 0)$
16: Get evolutionary equilibrium at iteration ϕ
17: Finalize population share as: $y_e^* \leftarrow y_e(\cdot)$
18: Evaluate final utility value W_e^* using Equation (15) at evolutionary iteration ϕ
19: **return** y^*, W^*;

constant time $O(1)$; thus, the entire block from lines 6-10 takes $O(|\mathcal{S}|)$ time complexity for all switches. Evaluating an average utility for switches in line 11 takes $O(|\mathcal{S}|)$ time complexity. Evaluation and updating of the population share are assumed $O(1)$ per switch; thus, the entire block from lines 12-15 takes $O(|\mathcal{S}|)$ time complexity for all switches. Hence, the overall time complexity of the Algorithm 1 is $O(\omega|\mathcal{S}|)$. Similarly, the overall time complexity of the Algorithm 2 $O(\phi|\mathcal{E}|)$ for all the edge nodes $\mathcal{E}$, where ϕ is an evolutionary stable state for Algorithm 2.

5 | PERFORMANCE ANALYSIS

We consider Mininet[#] the most suitable network emulation tool for our proposed SDEN architecture. Mininet provides a realistic virtual environment for custom topology and an SDN controller with OpenFlow compatibility. In the subsequent section, we discuss experimental setup, performance metrics, and results for our proposed D-RESIN algorithms in detail.

5.1 | Experimental Setup

The proposed SDEN network architecture is experimentally set up according to Table 3. Table 3 provides detailed information on the hardware, memory type, Mininet emulator, SDN switches, and SDN controller, along with their respective versions. To evaluate the performance of the proposed D-RESIN algorithm, we designed a multi-switch custom-based typology with varying Open vSwitches[‖], IoT devices, and Edge nodes as depicted in Table 4. The network topology has varying numbers of IoT devices

[#] https://mininet.org/
[‖] https://www.openvswitch.org/

(50, 100, 200), Open vSwitches (2, 5, 10), and edge nodes (10, 20, 30) to analyze performance under different network scales. This Mininet topology is controlled by the single Ryu SDN controller**. The detailed emulation parameters are shown in Table 4.

In this work, we consider the following parameters for simulation:

1. For the resource orchestration at the edge node, we consider an initial energy of 20 joules[21] for each edge node. The edge node consumes 50 nj/bit[28] for incoming data traffic for computation.
2. The processing rate is considered 0.35 flows per second at both the access and edge tiers.
3. Ethernet is considered for networking standard; hence, we have taken 1518 Bytes for frame size.
4. The evolutionary control factors are set as α = 0.01 at the access tier and β = 0.1 at the edge tier.

TABLE 3 Experimental Setup

Hardware	Intel® Core™ i7-9700 CPU @3.00GHz × 8
Operating System	Ubuntu 20.04.6 LTS
RAM	24 GB DDR4
Disk Space	1.0 TB
Network Emulator	Mininet (Version 2.31b1)
SDN Controller	Ryu Controller (Version ryu 4.34)
SDN Switch	Open vSwitch (Version ovs-vsctl 2.13.8)
Network Interface Standard	Ethernet
Programming Language	Python3 (Version 3.8.10)
Benchmarks	T-RESIN, FlowMan, RandomFlow

TABLE 4 Simulation Parameter

Parameter	Value
Number of Switches in Mininet Topology	2, 5, 10
Number of Edge Nodes in Mininet Topology	10, 20, 30
Number of IoT Devices in Mininet Topology	50, 100, 200
Initial Energy of Each Edge Node	20 Joule[21]
Network Energy Consumption	50 nJ/bit[28]
Ethernet Frame Size	1518 Byte[29]
Processing Rate (μ_s, μ_e)	0.35 flows per second
Evolutionary Control Factor	α = 0.01, β = 0.1

5.2 | Benchmarks

We assess D-RESIN Algorithm 1 by comparing it with competing schemes — T-RESIN, FlowMan, and RandomFlow schemes. Additionally, D-RESIN Algorithm 2 is evaluated against T-RESIN and RandomFlow schemes. In RandomFlow, the data traffic is randomized at both access and edge tiers, leading to unpredictable outcomes in network throughput, computational delay, and energy consumption within the SDENs network. Alternatively, FlowMan[15] strategically placed flow rules to establish a trade-off between throughput and delay in two-tier SDN architecture. FlowMan is not specifically tailored for SDEN networks; it does not address resource allocation at the edge tier, which is a critical component for optimizing performance in these environments. On the other hand, In T-RESIN, Agrawal *et al.*[21] have optimally allocated resources to achieve high throughput and data flows, contributing to a sustainable SDEN network. However, T-RESIN is unsuitable for meeting the SDEN network's delay-sensitive requirements.

** https://ryu-sdn.org/

5.3 | Performance Metrics

- *Average processing delay at switch:* It is calculated as the ratio of the total volume of data and total processing time of all the flows associated with the particular switch. It depends on the flow's processing rate as well.
- *Per-Switch Flow:* Per-Switch Flow is calculated as the number of flow rules associated with TCAM memory for each switch.
- *Per-Switch Throughput:* Per-switch throughput is calculated as the average amount of data processed for each SDN switch individually. It also depends on the flow association of each switch in the SDEN.
- *Average processing delay at edge node:* The average processing delay at an edge refers to the time spent processing the data flows after arrival. This metric is crucial for assessing the performance and efficiency of SDEN.
- *Average energy consumption at edge node:* Each edge node serves the data traffic generated from IoT devices. Average energy consumption is the ratio of total energy consumed in the SDEN network and the number of edge nodes.

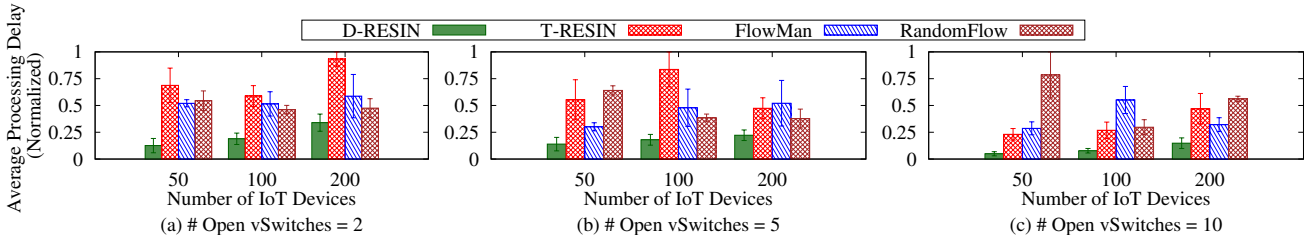

FIGURE 3 Average processing delay at switch

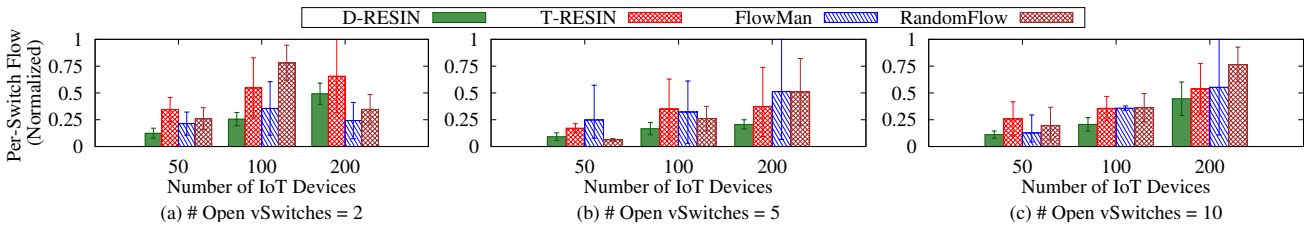

FIGURE 4 Per-switch flow

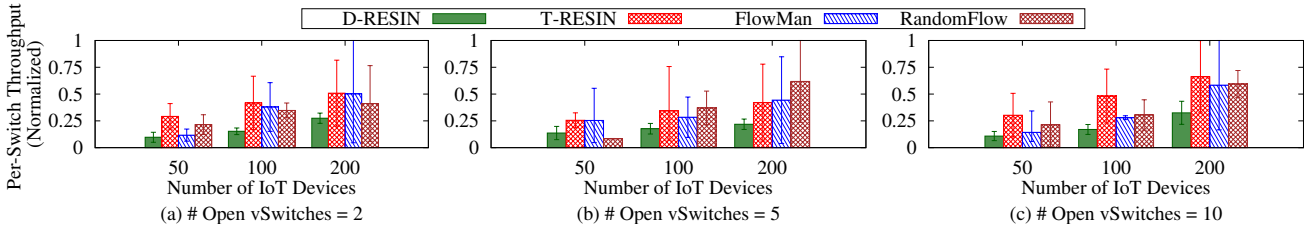

FIGURE 5 Per-switch throughput

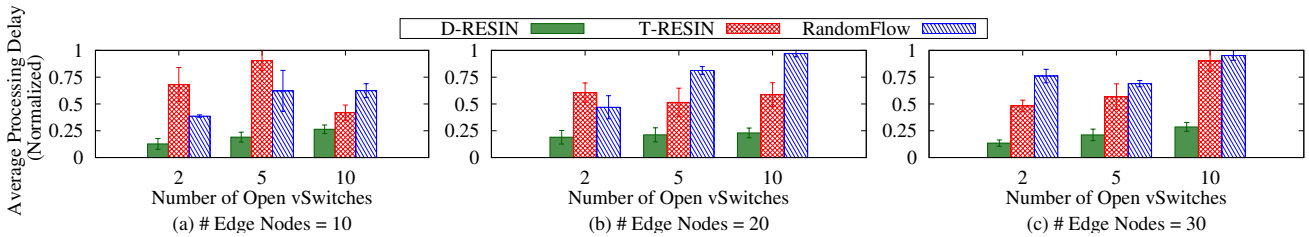

FIGURE 6 Average processing delay at edge node

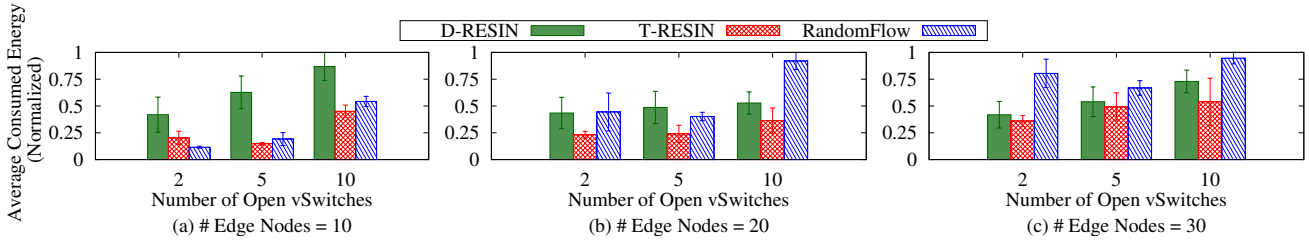

FIGURE 7 Average energy consumption at edge node

5.4 | Result and Discussion

This section presents the yield results from emulating Algorithm 1 at the Access tier and Algorithm 2 at the Edge tier within the proposed SDEN architecture. We evaluated the performance of our scheme under various topological configurations, including different numbers of Open vSwitches (2, 5, 10), IoT devices (50, 100, 200), and Edge Nodes (10, 20, 30), and compared these results to an existing competing scheme. The delay reduction percentages vary across different scenarios based on factors such as network topology, data traffic distribution, and resource allocation strategies. It also highlights the impact of evolutionary game theory-based dynamic resource orchestration in optimizing network performance. For each performance metric, we conducted 20 iterations per topology and calculated the variance of these results at the 95% confidence intervals. The values in the result sections are normalized and scaled to a standard range, likely between 0 and 1, for easier comparison across different scenarios.

From Figure 3, we observe that the average processing delay reduces as the number of switches increases for the D-RESIN topology since data traffic is distributed among switches based on optimized processing delay. Using D-RESIN, the average processing delay at the access tier decreases by 52.43-88.82%, 32.71-87.91%, and 25.50-94.76% than using the existing schemes— T-RESIN, FlowMan, and RandomFlow, respectively. However, T-RESIN does not consider the delay parameter for the mathematical modeling of SDEN architecture at all. FlowMan also focuses on data management, specifically addressing scenarios of very high or very low data traffic, and in RandomFlow, data traffic is very random among switches. Figure 4 shows D-RESIN performance in handling the flows per switch as the number of IoT devices increases. Compared to the existing scheme, D-RESIN associates the data flows with SDN switches, ensuring that every flow should be processed optimally. Figure 5 provides insights into the D-RESIN switch throughput performance, crucial for assessing its capability to manage traffic as the number of connected devices grows for delay-sensitive applications.

On the other hand, from Figure 6, we observe that using D-RESIN, the average processing delay at the edge tier decreases by 35.44-85.10%, 55.41-84.89% than using the existing schemes— T-RESIN and RandomFlow, respectively. This is because variation in the processing delay incurred at the edge nodes appears random for the T-RESIN scheme, and the data association is random for the RandomFlow scheme at the edge tier. However, Figure 7 depicts the efficiency of D-RESIN at the edge tier regarding energy consumption as the network configuration scales with an increasing number of Open vSwitches. As the number of switches and edge nodes increases, D-RESIN effectively distributes data traffic and optimizes resource allocation to reduce average processing delay at both the access and edge tiers. The D-RESIN framework demonstrates improved scalability even in larger and more complex network environments. Hence, we conclude that D-RESIN ensures a deduction in average processing delay at access and edge tier for the proposed SDEN architecture.

6 | CONCLUSION

In this paper, we proposed a novel framework, named D-RESIN, that addresses the challenges of dynamic resource orchestration in SDEN, ensuring high QoS for delay-sensitive applications. We leveraged the evolutionary game theoretic approach to ensure optimal processing delays between IoT devices, Open vSwitches, and edge nodes managed by the Ryu SDN controller. We demonstrated the existence of evolutionary equilibrium for D-RESIN at both access and edge tiers by proof. Our implementation and evaluation using the Mininet emulation environment showed that D-RESIN significantly reduces computation delay at both access and edge tiers to enhance the scalability and efficiency of SDEN. D-RESIN optimizes resource allocation in various IoT scenarios, such as smart traffic management, industrial IoT, and real-time healthcare monitoring, by dynamically adjusting network resources using an evolutionary game-theoretic approach.

This work can be extended by incorporating energy consumption for computation and ensuring energy-efficient resource allocation for SDEN. We plan to extend D-RESIN to incorporate priority-based application management for the IoT application based on their urgency and resource requirements. We also plan to integrate edge-cloud computing platforms to leverage the computational power and storage capabilities of cloud resources as well in the presence of mobile nodes. As a future enhancement, the management system can be improved to handle high device mobility and intermittent connectivity by integrating intelligent handover mechanisms and predictive mobility models. Leveraging AI-driven traffic forecasting and adaptive resource allocation can ensure seamless connectivity and reduce disruptions for SDEN. This will enhance the network's resilience and efficiency in dynamic IoT environments.

ACKNOWLEDGEMENT

This work is partially supported by DIStributed and Trustworthy Edge-Cloud Architecture for Future 6G-Enabled Healthcare (DISTECH-6G) project sponsored by the Finnish Ministry of Education and Culture. Ayan Mondal also acknowledges that this work was partially supported by the IIT Indore Young Faculty Research Seed Grant (YFRSG) Scheme (Grant No: IITI/YFRSG/2022-23/12).

References

1. Siow E, Tiropanis T, Hall W. Analytics for the Internet of Things: A Survey. *ACM Computing Survey.* 2019;51(4):1-36. doi: https://doi.org/10.1145/3204947

2. Agrawal L, Tiwari N. A Review on IoT Security Architecture: Attacks, Protocols, Trust Management Issues, and Elliptic Curve Cryptography. In: Social Networking and Computational Intelligence. Springer, Singapore 2020:457-465.

3. Stergiou C, Psannis KE, Kim BG, Gupta B. Secure integration of IoT and Cloud Computing. *Future Generation Computer Systems.* 2018;78:964–975.

4. Kumar T, Partala J, Nguyen T, Agrawal L, Mondal A, Kumar A, Ahmad I, Peltonen E, Pirttikangas S, Harjula E . Secure Edge Intelligence in the 6G Era. In: Security and Privacy for 6G Massive IoT. Wiley 2024.

5. Tefera G, She K, Shelke M, Ahmed A. Decentralized adaptive resource-aware computation offloading & caching for multi-access edge computing networks. *Sustainable Computing: Informatics and Systems.* 2021;30.

6. Alam I, Sharif K, Li F, Latif Z, arim MMK, Biswas S, Nour B, Wang Y, Wang Y, Wang Y . A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV. *ACM Computing Survey.* 2020;53. doi: https://doi.org/10.1145/3379444

7. Amadeo M, Campolo C, Ruggeri G, Molinaro A, Iera A. SDN-Managed Provisioning of Named Computing Services in Edge Infrastructures. *IEEE Transactions on Network and Service Management.* 2019;16:1464-1478.

8. Jazaeri SS, Jabbehdari S, Asghari P, Javadi HHS. Edge computing in SDN-IoT networks: a systematic review of numbers, challenges and solutions. *Cluster Computing.* 2021;24:3187–3228.

9. Liao Z, Pang X, Jingyu Zhang M, Xiong B, Wang J. Blockchain on Security and Forensics Management in Edge Computing for IoT: A Comprehensive Survey. *IEEE Transactions on Network and Service Management.* 2021;19(2):1159-1175. doi: https://doi.org/10.1109/TNSM.2021.3122147

10. Laroui M, Nour B, Moungla H, Cherif MA, Afifi H, Guizani M. Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications.* 2021;180:210-231. doi: https://doi.org/10.1016/j.comcom.2021.09.003

11. Bruschi R, Davoli F, Lago P, Lombardo A, Lombardo C, Rametta C, Schembra G, Schembra G, Schembra G, Schembra G . An SDN/NFV Platform for Personal Cloud Services. *IEEE Transactions on Network and Service Management.* 2017;14(4):1143-1156.

12. Wu Q, Wang S, Ge H, Fan P, Fan Q, Letaief KB. Delay-Sensitive Task Offloading in Vehicular Fog Computing-Assisted Platoons. *IEEE Transactions on Network and Service Management.* 2023;21(2):2012-2026.

13. Kim M, Hyeon D, Paek J, Kalla A. eTAS: Enhanced Time-Aware Shaper for Supporting Nonisochronous Emergency Traffic in Time-Sensitive Networks. *IEEE Internet of Things Journal.* 2022;9(13):10480-10491. doi: https://doi.org/10.1109/JIOT.2021.3124508

14. Balasubramanian V, Aloqaily M, Reisslein M. An SDN architecture for time sensitive industrial IoT. *Computer Networks.* 2021;186. doi: https://doi.org/10.1016/j.comnet.2020.107739

15. Mondal A, Misra S. FlowMan: QoS-Aware Dynamic Data Flow Management in Software-Defined Networks. *IEEE Journal on Selected Areas in Communications.* 2020;38(7):1366-1373.

16. Zhang X, Zhu Q. Scalable Virtualization and Offloading-Based Software-Defined Architecture for Heterogeneous Statistical QoS Provisioning Over 5G Multimedia Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications.* 2018;36(12):2787-28048.

17. Mondal A, Misra S, Maity I. Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks. *IEEE Systems Journal.* 2019;13(2):1359-1366. doi: https://doi.org/10.1109/JSYST.2018.2820745

18. Saha N, Bera S, Misra S. Sway: Traffic-Aware QoS Routing in Software-Defined IoT. *IEEE Transactions on Emerging Topics in Computing.* 2021;9(1):390-401. doi: https://doi.org/10.1109/TETC.2018.2847296

19. Xavier TCS, Delicato FC, Pires PF, Amorim CL, Li W, Zomaya A. Managing Heterogeneous and Time-Sensitive IoT Applications through Collaborative and Energy-Aware Resource Allocation. *ACM Transactions on Internet of Things.* 2022;3(2):1-28. doi: https://doi.org/10.1145/3488248

20. Jain DK, Tyagi SKS, Neelakandan S, Prakash M, Natrayan L. Metaheuristic Optimization-Based Resource Allocation Technique for Cybertwin-Driven 6G on IoE Environment. *IEEE Transactions on Industrial Informatics.* 2022;18.

21. Agrawal L, Mondal A, Obaidat MS. T-RESIN: Throughput-Aware Dynamic Resource Orchestration for IoE-Enabled Software-Defined Edge Networks. *International Journal of Communication Systems, Wiley.* April 2024;37(12). doi: https://doi.org/10.1002/dac.5802

22. Liyanagea M, Porambageb P, Dingc AY, Kalla A. Driving forces for Multi-Access Edge Computing (MEC) IoT integration in 5G. *ICT Express.* 2021;7(2):127-137. doi: https://doi.org/10.1016/j.icte.2021.05.007

23. Manogaran G, Rawal BS. An Efficient Resource Allocation Scheme With Optimal Node Placement in IoT-Fog-Cloud Architecture. *IEEE Sensors Journal.* 2021;21.

24. Sami H, Otrok H, Bentahar J, Mourad A. AI-Based Resource Provisioning of IoE Services in 6G: A Deep Reinforcement Learning Approach. *IEEE Transactions on Network and Service Management.* 2021;18.

25. Han Z, Niyato D, Saad W, Basar T, Hjørungnes A. *Game Theory in Wireless and Communication Networks*. Cambridge University Press, New York, 2012.

26. Williams HP. *Logic and Integer Programming*. International Series in Operations Research & Management ScienceBoston, MA: Springer, US, March 2009.

27. Drexl A. A simulated annealing approach to the multiconstraint zero-one knapsack problem. *Computing.* March 1988;40(1):1-8.

28. Heinzelman WR, Chandrakasan A, Balakrishnan H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: 2000; Hawaii, USA:1–10.

29. Zhao Z, Qiu Z, Pan W, Li H, Zheng L, Gao Y. Design and implementation of a frame preemption model without guard bands for time-sensitive networking. *Computer Networks.* 2024;243. doi: https://doi.org/10.1016/j.comnet.2024.110285