Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057755

ARTICLE

A Latency-Aware and Fault-Tolerant Framework for Resource Scheduling and Data Management in Fog-Enabled Smart City Transportation Systems

Ibrar Afzal[1], **Noor ul Amin**[1,*], **Zulfiqar Ahmad**[1,*] **and Abdulmohsen Algarni**[2]

[1]Department of Computer Science and Information Technology, Hazara University, Mansehra, 21300, Pakistan

[2]Department of Computer Science, King Khalid University, Abha, 61421, Saudi Arabia

*Corresponding Authors: Noor ul Amin. Email: namin@hu.edu.pk; Zulfiqar Ahmad. Email: zulfiqarahmad@hu.edu.pk

Received: 27 August 2024 Accepted: 19 November 2024 Published: 03 January 2025

ABSTRACT

The deployment of the Internet of Things (IoT) with smart sensors has facilitated the emergence of fog computing as an important technology for delivering services to smart environments such as campuses, smart cities, and smart transportation systems. Fog computing tackles a range of challenges, including processing, storage, bandwidth, latency, and reliability, by locally distributing secure information through end nodes. Consisting of endpoints, fog nodes, and back-end cloud infrastructure, it provides advanced capabilities beyond traditional cloud computing. In smart environments, particularly within smart city transportation systems, the abundance of devices and nodes poses significant challenges related to power consumption and system reliability. To address the challenges of latency, energy consumption, and fault tolerance in these environments, this paper proposes a latency-aware, fault-tolerant framework for resource scheduling and data management, referred to as the FORD framework, for smart cities in fog environments. This framework is designed to meet the demands of time-sensitive applications, such as those in smart transportation systems. The FORD framework incorporates latency-aware resource scheduling to optimize task execution in smart city environments, leveraging resources from both fog and cloud environments. Through simulation-based executions, tasks are allocated to the nearest available nodes with minimum latency. In the event of execution failure, a fault-tolerant mechanism is employed to ensure the successful completion of tasks. Upon successful execution, data is efficiently stored in the cloud data center, ensuring data integrity and reliability within the smart city ecosystem.

KEYWORDS

Fog computing; smart cities; smart transportation; data management; fault tolerance; resource scheduling

1 Introduction

Fog computing is an emerging technology that offers services to smart environments including smart cities, smart transportation, and smart campuses that deploy Internet of Things (IoT) smart sensors. Fog computing uses end nodes to distribute safe information locally while providing complete device authentication. It is a developed kind of cloud computing that includes improvements to deal with bandwidth, privacy, latency, storage, response time, security, and computation challenges. Three main components make up fog computing: end devices, fog nodes, and back-end cloud infrastructure

[1]. The IoT notion is supported by the advancement of fog computing in which almost all nodes and devices interact with one another. Fog computing offers its services to several IoT-based linked devices and applications such as smartphones, gadgets, and Google Glass [2,3].

Changes in the everyday use of smart services have led to significant advancements in computer networks and the development of the IoT concept. The IoT is a smart communications environment in which smart devices work as objects or "things" that can connect to one another, perceive their surroundings, and share data online. One trillion IP addresses or things were supposed to be linked to the Internet by 2022 as IoT networks [3]. Sensors in intelligent surroundings cooperate to carry out tasks. IPv6, wireless communication methods, and wireless sensors all contribute to the growth of smart surroundings. These configurations are available in many different configurations, such as smart cities, smart households, smart logistics, smart industries, and smart healthcare. The effectiveness of smart environments is improved when interconnected Internet of Things technologies and smart surroundings collaborate. A smart city is the result of the development of smart environments in metropolitan regions [4]. An effective concept of a smart city built on an IoT system is the Padova Smart City in Italy [5]. The IoT and smart environments are enabled by advances in a wide range of technical areas with a variety of IoT-based networking solutions. These solutions can be categorized as i) general-purpose "Constrained-Node Network" (CNN) technologies, and ii) "Radio-Frequency Identification" (RFID), which is primarily used for object and device identification. Smart lighting systems are significant applications of smart cities that provide automatic lighting systems in smart cities, which also optimize the utilization of power consumption [6–8].

Finding the best IoT technologies and solutions for a given smart environment can be challenging because there are so many high heterogeneities in terms of features and performance trade-offs. While all intelligent settings gather, analyze, and act on information, the scales at which they accomplish these things vary. Various vertical domains (such as smart transport, homes, health, cities, or factories) have varied needs, which affects technology choices and the strategies for where and how data is computed and how to use the information in a particular context [9–11]. IoT consists of complex settings with several heterogeneous components. Processing and storage resources will be under tremendous pressure to convert the vast amounts of data generated by sensor-equipped real-world objects into usable information or services. While some applications need complicated processing, such as time series analysis and historical data, others would be latency-sensitive. Latency refers to the time delay between the initiation of data and receipt of the data. In smart cities, transportation systems include different components such as sensors, cameras, vehicles, traffic management systems, and communication networks which are connected and exchange data [12,13]. Low latency is important in such a type of communication to ensure timely decision-making and response. It is measured in milliseconds (ms) and its acceptable value is not available as a standard value, but it varies depending on the nature of the application. For example, in the case of autonomous vehicles and collision avoidance systems, the range of a few milliseconds latency is acceptable so that vehicles can react to changing road conditions and avoid accidents [14]. Similarly, in the case of a traffic management system, a latency of a few seconds is acceptable but lower latency is always better. Given the typical resource limitations of IoT devices, it is therefore impossible to install a real-world IoT environment without a cloud/fog infrastructure [15–18].

However, when fog computing is selected, the challenge of job scheduling and resource management appears different and may be summed up as the following two points [19–22]:

- Each IoT device can always have numerous accessible fog nodes. The delay of data transfer to various fog nodes varies because of varied data transmission distances. Therefore, when deciding which fog node should be assigned to the task request, unequal data transmission delay must be considered. Such delays may create a fundamental problem for time-sensitive tasks in smart environments, such as accidents in smart transportation or disaster management, fire detection and prevention, and emergency healthcare services in smart cities. When the task request is planned to assign a fog resource that is nearby, there will be reduced costs associated with scheduling the data transfer.

- The devices in smart environments work in various conditions such as open air, closed environments, mobility-aware situations, and cold and hot weather conditions. There are chances of failure of nodes and task executions which can cause irreparable losses, especially for time-sensitive applications in smart cities such as accident or disaster management, fire detection and prevention, and emergency healthcare services. Thus, fault-tolerant resource scheduling and data management would be required in smart cities in such situations.

Moving data processing, including the generation of alarms with the help of raw data streams to fog nodes, is frequently important in many IoT applications since it lowers network delays and conserves a significant volume of network capacity. In some situations, the computational power available at the fog node may not be adequate, requiring a data processing transfer from the fog environment to the cloud infrastructure. Depending on available computational resources on the fog and cloud side, this sort of data processing location must be chosen at runtime. As a result, the following issues must be resolved:

1. To enable flexible placement of data computation across the clusters, a single framework-based solution is needed for processing, portability, and administrative convenience.
2. It is necessary to implement a fault-tolerant mechanism locally along with a replication mechanism for the provision of backup nodes to maintain the system state locally at the fog nodes.

In fact, the data may be iteratively processed and supplied to another node since the processing steps collect duplicated data from any accessible node. It causes the system to perform unreliable behaviours, especially in safety-critical contexts, such as when an alarm is generated or when a turbine is being controlled. At other times, data might not have been processed, which could cause the program to freeze at any time. In order to administer large clusters of cloud/fog through a single interface, a federated management system must be developed, regardless of the hardware [23,24]. It is possible for data to be obtained from a large variety of available structures by an IoT network made up of several IoT nodes, controller nodes, or actuator nodes. However, when the working efficiency of IoT nodes is reduced, the data they collect is inaccurate and leads to bad conclusions. Therefore, it is essential to increase the capacity of an IoT network to identify IoT nodes that are not working properly [25].

Briefly, the importance of smart environments such as smart city transportation cannot be denied in the modern world. The big picture in the context of this study is shown in Fig. 1.

A smart city is a conceptual term for an urban environment that utilizes advanced technologies to improve the quality of life. Common smart city applications include smart hospitality, smart communities, smart grid, smart factories, smart warehouses, smart healthcare and smart transportation systems [26,27]. The goal of this research is not to build a smart city or a broad application of a smart city per se. Instead, this study is focused on smart transportation, and the challenges of smart transportation regarding latency, fault tolerance, and data management.

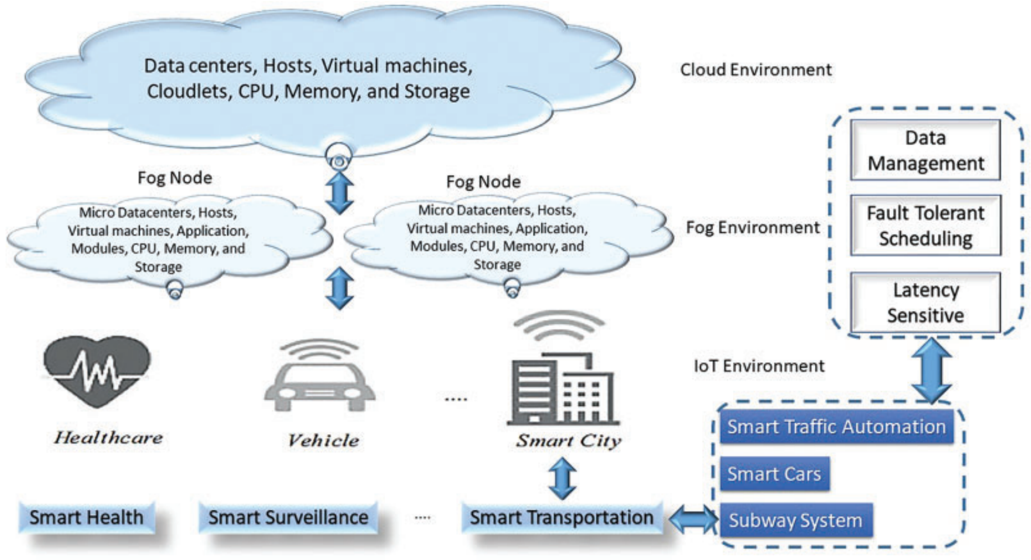

Data centers, Hosts, Virtual machines, Cloudlets, CPU, Memory, and Storage
Cloud Environment
Fog Node
Micro Datacenters, Hosts, Virtual machines, Application, Modules, CPU, Memory, and Storage
Fog Node
Micro Datacenters, Hosts, Virtual machines, Application, Modules, CPU, Memory, and Storage
Fog Environment
Data Management
Fault Tolerant Scheduling
Latency Sensitive
IoT Environment
Healthcare
Vehicle
....
Smart City
Smart Traffic Automation
Smart Cars
Subway System
Smart Health
Smart Surveillance
....
Smart Transportation

Figure 1: Smart city transportation in a Fog/Cloud environment

Smart environments work with smart devices such as sensor nodes, data collection nodes, data storage nodes, and computational nodes. Large volumes of data are being generated and processed by these environments. A single computing paradigm such as a cloud or fog is not enough for scheduling tasks and management of data generated by smart environments. There are time-sensitive applications in such environments such as accident prevention in smart transportation or disaster management, fire detection and prevention, and emergency healthcare services. Maximum power consumption, failure of nodes and tasks, and delay in processing such applications would result in loss of human life and financial assets. Therefore, to address the above-mentioned challenges, this paper presents a fog-cloud-based solution for resource scheduling and data management with latency and fault tolerance. Accordingly, the paper will present a latency-aware and fault-tolerant Resource Scheduling and Data Management system for Smart City Transportation in a Fog Environment.

The research questions of the study are as follows:

1. How can a resource-scheduling framework for latency-sensitive applications be designed for smart city transportation systems?
2. How can fault tolerance be implemented in a latency-aware resource-scheduling framework for a smart city transportation system in a fog environment?
3. How can a data management technique be designed in a latency-aware resource-scheduling framework for a smart city transportation system in a fog environment?

There are also three objectives in this research:

1. To design and implement a latency-aware resource-scheduling framework for smart city transportation system that prioritizes and executes tasks by considering latency during execution.
2. To implement a fault-tolerant mechanism in a latency-aware resource-scheduling framework for smart city transportation systems.
3. To implement a data management technique in a latency-aware resource-scheduling framework for a smart city transportation system.

The rest of the paper is organized as follows: Section 2 presents a literature review. Section 3 provides the System Design and Model. Section 4 provides performance evaluation. Section 5 presents results and discussions and finally, Section 6 concludes the paper.

2 Literature Review

The literature is reviewed for resource scheduling and fault-tolerant techniques in smart environments. The literature review was categorized into three parts, i.e., latency-aware resource scheduling summarized in Table 1, fault-tolerant techniques summarized in Table 2, and data management techniques for smart environments summarized in Table 3.

There are several frameworks for designing and modelling systems that have been developed to suit the smart city infrastructure [28–30]. In [28], the authors propose an Intelligent Priority Selection (IPS) mechanism using superior mathematical operators for Metro-owned charging stations (MCSs). To manage parameter uncertainty, the Unscented Transform (UT) was used, and simulation results show that the proposed IPS has a CPU time of 75% compared with other existing meta-heuristic methods in literature. In [29], a health monitoring and diagnostic framework was proposed to monitor the geo-distributed edge clusters, which process large amounts of data produced by smart city applications. This framework is founded on the MapReduce paradigm for distributed big data processing across edge clusters located in the smart city. In this system, the SmartMonit monitoring agent gathers health statistics from the edge devices and forecasts future failures of devices using an artificial neural network-based self-organizing map. The framework was then implemented across different clusters to perform failure detection on the framework. The research in [30] discusses intelligent energy management solutions using various mechanisms to respond to increased energy demand and exhaustion of resources, which in turn cause increased energy consumption and maintenance of building infrastructures. The raw data acquired is employed for overseeing the system, regulating it, and increasing the general effectiveness of the system.

For the fog environment, the authors in [31] propose a latency-aware strategy for application modules that can handle the various delay and data signal processing demands of different applications. The goal of the policy is to optimize resource use in the fog environment and ensure applications' "Quality of Service" (QoS) in meeting service delivery deadlines. The authors of [31] use the fog simulation provided by the iFogSim to model and assess the suggested policy in this paper. The simulation experiment findings show a noticeable increase in performance compared to other latency-aware techniques. The study in [32] propose a revolutionary method for managing and allocating resources in order to assure resource use at the fog layer. The "Technique for Resource Allocation and Management" (TRAM), a mechanism for resource management, was suggested. Using the expectation maximization (EM) algorithm, this method tracks the degree of job intensity and determines the current resource situation. A wireless system was used to manage all the resources. For fog computing, the study [32] offers a scheduling method for the resource grading procedure. This method's effectiveness was evaluated using the iFogSim simulator, and the outcomes were contrasted with First Come First Served (FCFS), Shortest Job First (SJF), and Modified Particle Swarm Optimization (MPSO) algorithms. The experimental findings showed that TRAM efficiently reduces task execution time, network usage, energy use, and average loop delay.

In [33], a fog/cloud architecture AdaptiveFog was suggested as a solution to deal with issues with wireless access latency in various Long-Term Evolution (LTE) networks. The main goal was to raise customer confidence in the service. To measure the performance disparity between various LTE networks, Kantorovich-Rubinstein (K-R) distance was utilized instead of statistical distance. A smart vehicle can use KR distance to dynamically switch between Mobile Network Operator (MNO) networks, or mobile network operators, which provide fog and cloud services. According to the results, this improves fog and cloud latency confidence levels by 30% to 50%, respectively. For the purpose of comparing results, AdaptiveFog and myopic techniques were evaluated. Information about the K-R distance and switching costs at all driving locations is necessary for both techniques. Another issue was the pre-calculation of decision thresholds at the User Equipment (UE) or fog node. In [34], a framework involving vehicular networking clouds with IoT (VCoT) was proposed which reduces the problem of integrating IoT with the existing Vehicular Ad-hoc Network (VANET) system. VCoT used LoRaWAN-based vehicular networks for different real-world application scenarios. LoraWAN is low-power, long-range, and efficient for communication options. LoRaWAN-based IoT infrastructure used RSUs to offload information to the cloud and retrieve data back from it. IoT and VANET integration for VCoT has an issue with distinct specification standards.

A unique Multi-Class-Based Classification (MCBC) technique was put forth in [35] that would reduce the average response time. The Multi-layer Latency Aware Workload Assignment Strategy (MLAWAS) was a multi-layer approach used to distribute workload among various E-Transport applications. A Q-learning strategy was also employed, which determines performance at each stage of execution and has a lower migration latency compared with baseline approaches, to address the drawbacks of existing algorithms, which calculates application performance at initial and final steps.

Table 1 summarizes the studies under [31–35], comparing them in terms of contribution, advantages, limitation and simulation used.

Table 1: Latency-aware resource scheduling techniques in smart environments

Ref.	Contribution	Advantages	Limitations	Simulation
[31]	Heuristic and constrained-based optimization	Latency-aware application modules optimization of fog nodes	No real-world scenario	iFogSim
[32]	TRAM	Minimizes loop delay for latency-sensitive, execution time, network consumption, and energy consumption	For a smaller number of nodes, energy consumption was high	iFogSim
[33]	AdaptiveFog	Maximizes the service confidence level	Switching cost increased latency and optimizing for LTE were quite difficult	Mathematical modelling
[34]	VCoT	Lower Latency Priority-based communication mechanism	Not good for the Fog computing paradigm because data needs smarter pre-processing and also has heterogeneity in integrating VCoT	SUMO Android auto client

(Continued)

Table 1 (continued)

Ref.	Contribution	Advantages	Limitations	Simulation
[35]	MLAWAS	Latency and average response time minimized. It can adopt dynamic changes quickly and measure performance at each step	New framework based on the serverless model required	Edge-cloudsim and VANET++

Table 2: Fault-tolerant resource scheduling techniques in smart environments

Ref.	Proposed technique	Advantages	Limitations	Simulation
[20]	Fault-tolerant scheduling method (FTSM)	Appropriate fault-tolerant selection algorithm from existing (RM and CKM) for the core request. Reliability and capacity of cloud optimization.	Selecting the K value from the fault handler was a challenging task and hence decreased the overall efficiency of the system.	Real world modelling
[23]	IoTEF architecture	Manage hardware and network connectivity failures by formulating functional and non-functional requirements and implementing them on the Otaniemi3D project.	Edge devices and security at the edge-level is a big concern and limited functionality for mobile edge devices creates faults, which are not managed properly.	Real world modeling
[25]	Feedback model-based predictive approach	Algorithms provide feedback and a mechanism in which incorrect data were self-corrected.	Hot spot detection from an imprecise IoT node was difficult and created faults.	Markov chain with transition matrix
[36]	IoTSCA	Fault-tolerant fog computing platform with Service-oriented middleware in distributed smart city services was proposed.	Fog nodes suddenly fail to create faults and disrupt overall operations.	Mathematical modelling

(Continued)

Table 2 (continued)

Ref.	Proposed technique	Advantages	Limitations	Simulation
[37]	Fault Tolerant Data management (FTDM)	Categorization of nodes into task-based and node-based fault-tolerance and data management was proposed.	Only covers e-health application for Smart city fault and data management.	iFogSim
[38]	ESMA	A fault tolerant mechanism was proposed and detected faults in smart metrics to transfer data and the overall performance will not be affected.	Communication overhead is higher in ESMA.	Real world modelling

Table 3: Data management techniques for smart environments

Ref.	Proposed technique	Advantages	Limitations	Simulator
[37]	FTDM	Categorization of nodes into task-based and node-based fault-tolerance and data management.	Only covers e-health application for smart city fault and data management.	iFogSim
[39]	SSC-DLC	A new paradigm for smart city data management that decreases network data traffic.	Only data from Barcelona smart city was considered.	Real world modelling
[40]	ITCS	Architecture for data processing in smart cities and crowd management.	Advanced visualization tools were required for system efficiency.	Hadoop scheduler
[41]	CityPulse framework	It analyzed and scheduled data to provide real-time bus locations.	Just a tiny custom-made application which was specific for Brasov city.	Real world modelling
[42]	IoT-IPTS	IPTS was managed and obtained context data from various sources by using emergent intelligence techniques (EIT) and Multi-Agent Systems (MAS).	The increasing number of MAs generates a greater amount of connections overhead.	Real world modelling

IoT Edge Framework (IoTEF), a four-layer architecture, was proposed for managing federated edge and cloud fault-tolerant applications in [23]. The Otaniemi department's use case for smart buildings and the 3D project were both used for idea-proofing and assessment of IoTEF proficiencies. The main goals were lessening latency, conserving network capacity, and addressing hardware and network connectivity concerns. However, regardless of the hardware, a federated management system must be established in order to control huge cloud/fog clusters through a single interface. It is essential to advance the IoT network's capacity to identify IoT nodes that are not working properly [25]. Through consistent monitoring and control, faulty nodes that gathered improper data were searched and were self-corrected by replacing faulty sensors with new ones. As a result, data collected from the environment was retained constantly and as a result, this increased the maintenance and reliability of IoT. The proposed algorithm was offered to respond to improper data which can be self-corrected, but the capability of perceiving vague IoT nodes from a hot (cold) spot was limited.

On the basis of computational cores, time tolerance, and time sensitivity, a new strategy for scheduling fault-tolerant cloud/fog was proposed in [20]. Using a prepared executive list, applications with high time sensitivity were directly charted to one or more edge devices. Time-tolerant requests may be independently allocated to one or more cloud machines. Resources at the cloud core were allocated to core demands. The classifier selects the most appropriate fault-tolerant strategy among the Recovery Mechanism (RM), Checkpointing (CHK), and resubmission techniques to obtain fault tolerance, reliability, efficiency, and capacity of cloud optimization. The primary difficulty in achieving a service for applications with time-sensitive fault handlers was choosing the right K value, as choosing an incorrect K value would result in multiple copies of the application being sent, which would increase the cost of the cloud resources. Furthermore, the reliability and fault-tolerance concerns for fog platforms enabling IoT-based smart city applications was covered in [36]. A "service-oriented middleware" (SOM) was created and run on smart city services to achieve a high level of fault tolerance for fog computing. To monitor fog nodes, SmartCityWare considers all resources as a core service and offers fault-tolerant fog computing that supports IoT-based smart city applications (IoTSCA). Additionally, it determines the status of the resources and modifies broker activities for fault tolerance services that can be used.

In [37], fog node and task-based failures were handled for IoT devices in health care. Task and node-based failures were managed by a "Fault Tolerant Provider" component. One fog node from which a copy of a failed job was provided was kept available for task allocations. The "Decision Maker" found the errors. In case of failure, the patient's alarm and the patient's consultant's alarm were both activated. The decision-maker module was used to manage the patient's data and classify abnormal and normal data. The ifogSim was used to run the simulation, and it was compared to Greedy Knapsack Scheduling (GKS). The plan was specifically designed for the medical industry.

Next, an "Effective and Secure Multidimensional Data Aggregation" ESMA arranged and encrypted multidimensional data into a Paillier ciphertext before effectively decrypting it [38]. The Paillier cryptosystem was employed to safeguard secrecy in a fog environment, and the batch verification approach was used to establish reliable authentication. Moreover, even if smart meters were unable to deliver data, ESMA's fault tolerance ensures that the final aggregate result will not be affected. ESMA can be modified to answer queries other than data summing. The analysis demonstrates the scalability and affordability of ESMA for both computing and communication. For instance, 40 data types and 500 smart meters can both fit into a Paillier ciphertext with a 16-bit capacity for each data type. ESMA defends user privacy and withstands numerous security intrusions.

In [39], a fog to cloud architecture for data management was proposed. The authors managed resources from global fog to cloud environments. The major advantage of the proposed architecture was that it reduced the latency for sensitive applications with maximum utilization of computing facilities. In [40], an architecture for processing smart city data was proposed. The authors mention various applications of data processing in smart cities, including intelligent transportation systems, water resource management, crowd management, and noise and air pollution management. Based on these applications, the authors proposed an integrated flow-oriented data processing architecture. For implementation, they consider two case studies, i.e., intelligent transportation systems and crowd management systems.

In [41], a novel application based on CityPulse framework was developed for Brasov public Transportation Company which used data from Urban Resource and Bus Scheduling (URBUS) provider. The suggested system offers bus riders route suggestions, incident alerts, and real-time information on the status of the buses that are passing from bus stations. The information about bus arrivals at stations and citizen-reported incidents is processed in real time to achieve this. The metadata for bus stations contains information about the stations' locations and other pertinent information. In [42], an intelligent public transportation system (PTS) called IoT-IPTS was introduced. It was called an intelligent system because it used context information of transportation entities to predict routes, alternative modes, roadside units, and departure times. Mobile agents using emergent intelligence techniques gave contextual information.

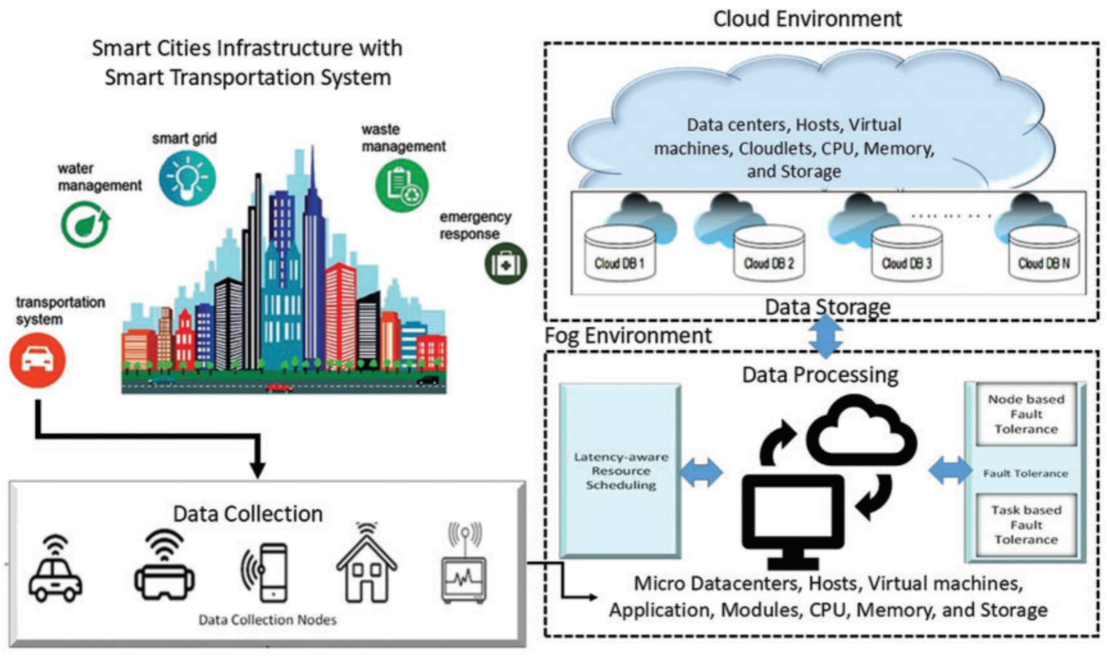
The work presented in [43] introduces a new framework for hierarchical edge computing in smart cities in which the resource allocation requirements are navigated for the increasing quality of service heterogeneity across the many connected devices. Using an attention mechanism, the approach learns and selects key features from large data generated at edge nodes and provides timely processing for the high-priority and delay-sensitive applications, including health care and industrial applications. The scheme integrates Q-learning in order to prioritize tasks and assigns resources efficiently and proactively in the edge network to enhance resource utilization, reduce the processing time of tasks, and ensure the quality of service of all the services. The study presented in [44] gives a detailed description of the IoT platform with an emphasis on the key components, structure, and communication system as well as the various applications. The authors talk about IoT as an enabler for smart city solutions with a focus on health, transportation, and smart grids. The authors also discuss the particularities of different communication protocols together with the existing potential and limitations. They propose significant problems that need to be solved in the process of IoT development, presenting such problems in the fields of security, architectural design, and compatibility, as well as describing potential scenarios for further IoT evolution.

The currently implemented resource management schemes schedule the resources and manage data in smart environments with the aim of reducing processing time and latency. However, there are various time-sensitive applications in smart environments, especially in smart cities, such as the smart transportation system. In smart transportation systems, there are applications for accident and disaster management, fire detection and prevention, and healthcare emergency services. These applications work with smart devices such as sensor nodes, data collection nodes, data storage nodes, and computational nodes. Maximum energy consumption, failure of nodes, and tasks, and delay in processing such applications would result in loss of human life and property. Therefore, to address the issues of energy consumption, delay and fault tolerance in smart cities, this paper proposes Latency and Fault-tolerant Resource Scheduling and Data Management for Smart Cities in a Fog Environment.

3 System Design and Model

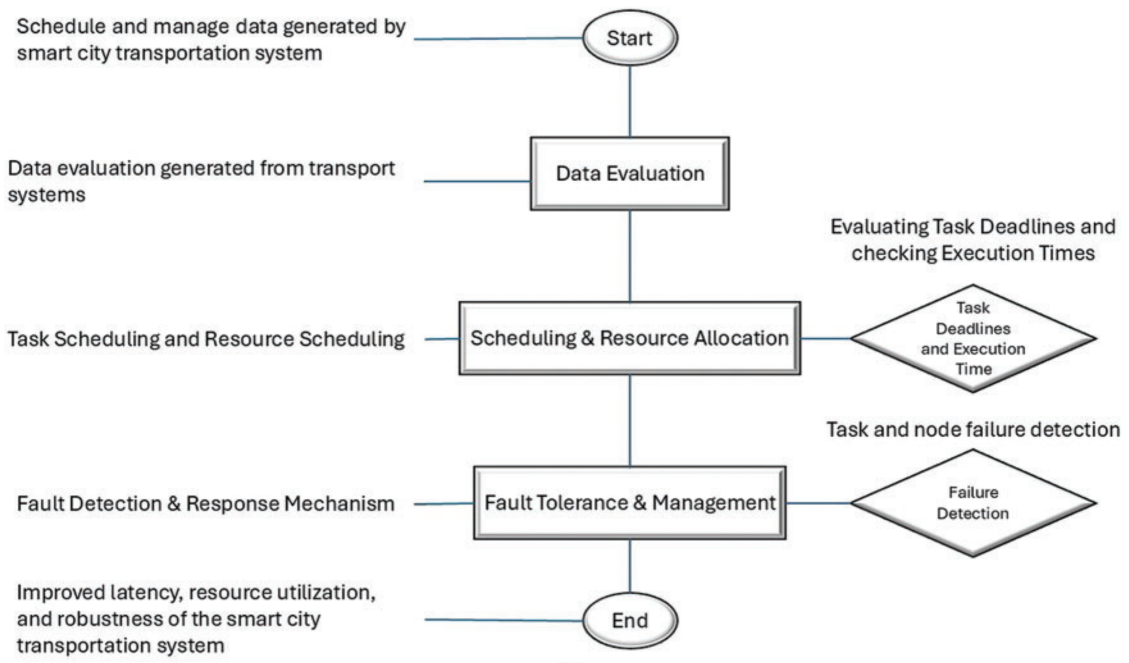
This research schedules and manages the data generated by smart city transportation systems with the integration of a fog environment by presenting a latency-aware and Fault-tolerant Resource Scheduling and Data Management (FORD) for Smart Cities in a Fog Environment as shown in Fig. 2a with the flowchart shown in Fig. 2b. The first stage includes the evaluation of data generation from various sources in the transport systems, focusing on latency and failure tolerance. The proposed framework seeks to employ fog computing to link cloud resources to reduce execution time by scheduling tasks to be executed on the nearest available node. In scheduling, the execution times of the task-resource pair (T_exe(i, j)), the deadlines (D(i)), and the availability of the resource (Rij) are considered in resource allocation. The objective function, as derived from this evaluation, represents a measure of latency, but with an optimization objective of fully utilizing available resources. In the second stage, the framework uses fog nodes for data analytics in real-time, which improves the efficiency and effectiveness of the complete system due to the brief time it takes to transmit data to distant cloud data centres. The fog nodes' integration is managed by a condition that limits the use of the cloud resources by fog nodes to execute tasks (T_fog $\leq$ T_cloud). The third stage is related to the implementation of fault tolerance in the integrated environment for handling node and task-based faults. If there is a node failure, it takes little time to redirect the tasks to other nodes with enough resources to avoid much loss. This process entails selecting an extra node of capability and redistributing the contingent tasks. In case of a task failure, the framework uses task redundancy for rescheduling the replica of the task on the original node or on any other node to ensure that the task is completed correctly. The total availability of the system is given based on the probability of node and task failures, which signifies the robustness of the framework in terms of constant operation.

There are two main components of the proposed framework:


Smart Cities Infrastructure with Smart Transportation System
smart grid
waste management
water management
emergency response
transportation system
Cloud Environment
Data centers, Hosts, Virtual machines, Cloudlets, CPU, Memory, and Storage
Cloud DB 1
Cloud DB 2
Cloud DB 3
Cloud DB N
Data Storage
Fog Environment
Data Processing
Latency-aware Resource Scheduling
Node based Fault Tolerance
Fault Tolerance
Task based Fault Tolerance
Data Collection
Data Collection Nodes
Micro Datacenters, Hosts, Virtual machines, Application, Modules, CPU, Memory, and Storage

(a)

Figure 2: (Continued)


Schedule and manage data generated by smart city transportation system
Start
Data evaluation generated from transport systems
Data Evaluation
Evaluating Task Deadlines and checking Execution Times
Task Scheduling and Resource Scheduling
Scheduling & Resource Allocation
Task Deadlines and Execution Time
Task and node failure detection
Fault Detection & Response Mechanism
Fault Tolerance & Management
Failure Detection
Improved latency, resource utilization, and robustness of the smart city transportation system
End

(b)

Figure 2: (a) An overview of the FORD strategy. (b) Flowchart for the proposed FORD strategy

a. Latency-Aware Resource Scheduling and Data Management

In smart city transportation systems, data processing and decision-making are the important parameters, and thus, latency-aware resource scheduling and data management play a pivotal role. The proposed framework ensures that the data generated from various sources within the transportation system are efficiently processed with minimum delay and makes sure that the resources are available from both fog and cloud environments. Resource scheduling involves the allocation of computing resources including CPU, memory, and storage to various kinds of tasks in such a way that it minimizes latency and maximizes efficiency. The proposed framework considers the time taken to execute tasks on different resources and aims to minimize execution time.

Let $T_{exe}(i, j)$ represents the execution time of task 'i' on resource 'j', $D(i)$ represents the deadline for task 'i', and R_{ij} is the availability of resource 'j'. To ensure that the tasks are completed within their deadlines and resources are utilized optimally, the objective function to minimize latency is represented by Eq. (1).

$$Minimize \sum_{i=1}^{N} T_{exe}(i, j) \tag{1}$$

The optimization function represented by Eq. (1) is subject to the condition given in Eqs. (2) and (3).

$$Minimize \sum_{j=1}^{M} T_{exe}(i, j) \leq D(i) \tag{2}$$

$$R_{ij} \geq 1 \tag{3}$$

where, N represents the total number of tasks, M is the total number of resources, and the constraints ensure that the tasks are completed within their deadlines and with available resources.

The data management component aims to optimize the storage and transfer of data while considering the available resources and minimizing latency. The large volume of data generated by smart city transportation systems is required to be managed efficiently. This involves storing, processing, and transferring data in a way that minimizes latency and ensures reliability. Let S denotes the total data size generated; B is the available bandwidth for data transfer, and L is the latency for data transfer. The objective function is to minimize latency subject to the condition given in Eq. (4).

$$\frac{S}{B} \le L \tag{4}$$

The condition given in Eq. (4) ensures that the time taken to transfer data does not exceed the available bandwidth, thus minimizing latency.

Fog computing extends cloud computing to the edge of the network, and it brings the computation and storage facility closer to the data source. The proposed framework utilizes the fog nodes for real-time processing of data generated by transportation systems and reduces the latency associated with transferring data to distant cloud data centres. The objective function of the integration of fog nodes is to minimize the execution time subject to the condition given in Eq. (5).

$$T_{fog} \le T_{cloud} \tag{5}$$

The condition given in Eq. (5) ensures that tasks are executed on fog nodes whenever possible to minimize latency, with the option of falling back to cloud resources if necessary.

Algorithm 1 deals with the data handling and job execution optimization that smart city transportation systems require to process data quickly and make decisions. It starts by taking in a range of inputs, such as assignments, resources, time constraints, data volumes, bandwidth, fog, and cloud latency. The main goal is to maximize efficiency and minimize latency by allocating tasks to resources in an efficient manner. The algorithm calculates the execution time for each task-resource combination by iterating through the available resources for each task. To guarantee that jobs are finished on time and with the resources available, it then assigns the task to the resource with the shortest execution time. The system then calculates overall latency depending on data size, available bandwidth, and latency criteria for fog and cloud environments to control data transmission latency. Tasks are carried out on either fog or cloud nodes to reduce overall latency, depending on how fog and cloud latency compare. The method also provides utility functions for task assignment to resources, task execution on fog or cloud nodes, and execution time calculation. The algorithm optimizes data management and resource scheduling in the transportation systems of smart cities, being the essential aspect for real-time data processing and decision-making.

Algorithm 1: Latency-aware resource scheduling and data management

```
1. Begin
2. Input: Tasks, resources, deadlines, data_size, bandwidth, fog_latency, cloud_latency
3. Output: Tasks execution on fog and cloud environments
4. Procedure: LatencyAwareResourceSchedulingAndDataManagement (tasks, resources, deadlines,
   data_size, bandwidth, fog_latency, cloud_latency)
```

(Continued)

Algorithm 1 (continued)

```
5. Resource Scheduling
  for each task ($T_i$)
      min_exe_time = infinity
      selected_resource = null
      for each resource ($R_j$)
          exe_time = CalculateExecutionTime ($T_i$, $R_j$)
          if exe_time < min_exe_time
          min_exec_time = exec_time
          selected_resource = $R_j$
          AssignTaskToResource ($T_i$, $R_j$) end if
      end for
  end for
6. Data Management
  total_latency = 0
  for each data ($D_i$) in data_size:
                      $transfer\_time = data/bandwidth$
      if ($transfer\_time <= fog\_latency$)
                      $total\_latency+ = fog\_latency$
      else
      total_latency += cloud_latency
      end if else
  end for
7. Integration with Fog Environemnt
  if fog_latency <= cloud_latency
     ExecuteTasksOnFogNodes()
     else
     ExecuteTasksOnCloudNodes()
     return total_latency
   end if else
8. CalculateExecutionTime ($T_i$, $R_j$)
   return T_exe ($T_i$, $R_j$)
9.  AssignTaskToResource ($T_i$, $R_j$)
10. ExecuteTasksOnFogNodes( )
11. ExecuteTasksOnCloudNodes( )
12. End
```

b. Fault Tolerant Implementation

Fault-tolerant mechanisms are implemented in the proposed framework by addressing both node-based and task-based failures. An alternate node is assigned for task execution to the failure of nodes and the failed tasks are re-executed by generating replicas of the task before execution.

When a node fails, the proposed framework ensures that tasks originally assigned to the failed node are rerouted to alternative nodes for execution. Let N denote the total number of nodes, and R_j denotes the set of resources available on node j. The following steps will be followed for the reassignment of tasks to alternative nodes, and it ensures that tasks affected by node failures are promptly rerouted to alternative nodes. This mechanism minimizes the downtime and ensures continuous operation.

1. *For each task T_i assigned to the failed node*
2. *Find an alternative node 'j' with sufficient resources to execute T_i*
3. *Reassign T_i to the alternative node 'j' for execution*

In case of task failure, the framework implements task replication to ensure successful execution. Upon detecting a failed task T_i, the framework uses a replica of T_i and schedules its execution on the availability of the same node or an alternative node.

The following steps are used to guarantee that the replicas of failed tasks are re-executed to achieve successful completion.

1. *If task T_i fails on node N_j*
2. *Retrieve replica of task T_i*
3. *Schedule execution of replica of task T_i on the same node or an alternative node*

Let F_{node} denote the probability of node failure and F_{task} denote the probability of task failure. The overall system availability A considering both node and task failures is calculated with the help of Eq. (6).

$$A = 1 - (1 - F_{node})^N \times (1 - F_{task})^T \tag{6}$$

where N represents the total number of nodes, T denotes the total number of tasks, F_{node} and F_{task} denote the probabilities of node and task failures, respectively.

4 Performance Evaluation

The proposed FORD strategy will be evaluated in the following steps:

1. A comprehensive latency-aware resource scheduling and data management framework has been designed for smart city transportation systems.
2. A fault-tolerant mechanism has been implemented in latency-aware resource scheduling frameworks.
3. The proposed FORD strategy is implemented in the iFogSim2 [45] simulator and following performance evaluation parameters [37,46].
    i. Execution Cost: The financial cost needs to be executed by the collection of tasks related to the smart city transportation system.
    ii. Latency: This is the overhead time taken to complete the sequence of tasks related to the smart city transportation system.
    iii. Execution Time: The duration needed to execute the collection of tasks related to the smart city transportation system.
    iv. Energy Consumption: The power spent by the devices and resources during the execution of a sequence of tasks related to smart city transportation systems.
4. The proposed FORD strategy was compared with the existing state-of-the-art strategy Crowd-Sensing_Microservices_RandomMobility_Clustering (CS-MRMC) [47].
5. The CDC (Crowd-sensed Data Collection) dataset was utilized. It is available online in the Github repository of iFogSim2 with the link: https://github.com/Cloudslab/iFogSim (accessed on 18 November 2024).

Crowd-sensing collects immense quantities of data from sensors connected to the internet, which can subsequently be analyzed in order to extract intricate information. The Crowd-sensed Data Collection (CDC) application facilitates urban road network planning through a mobile crowd-sensed scenario. The design of road networks and the management of traffic signals present formidable obstacles in urban environments [47]. These duties can therefore be enhanced through the implementation of sophisticated machine learning algorithms. Since precise decision-making necessitates vast quantities of data, vehicular crowd-sensing is implemented as a data collection method. Real-time location and speed data, which are detected and transmitted by sensors installed on mobile vehicles, can be utilized to ascertain the traffic conditions of road networks. By applying this approach, vehicles have the ability to willingly contribute data to data analytic platforms, leading to the accumulation of substantial quantities of data. By utilizing fog computing environments, these applications can process data in closer proximity to the periphery network, which consequently alleviates the strain on the data transmission networks that link sensors to the Cloud [47].

5 Results and Discussion

Simulation results are analyzed in respect of the performance evaluation parameters, i.e., execution cost, latency, execution time, and energy consumption. These parameters are considered the most relevant and important in the context of a smart city's transportation system.

5.1 Execution Cost

The simulation results show that the proposed FORD strategy significantly performed well as compared with the existing CS-MRMC strategy with respect to execution cost as shown in Fig. 3. The reason is that, in the proposed FORD strategy, due to the implementation of latency-aware resource scheduling and the provision of fault-tolerant mechanisms, the failed tasks are re-executed immediately upon failure, whereas, in the existing approach, the complete loop is required to be executed. Therefore, there is a minimum execution cost for the proposed FORD strategy.

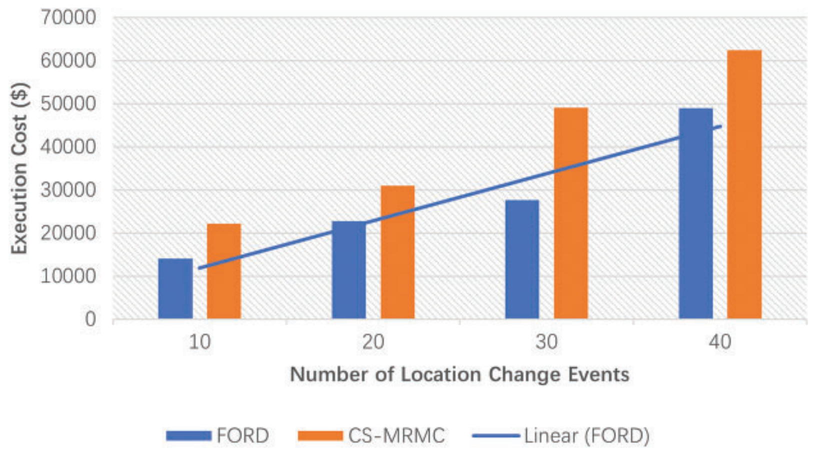


Figure 3: Execution cost comparison of proposed FORD strategy and CS-MRMC

5.2 Latency

The simulation results show that the proposed FORD strategy significantly reduces the latency as compared with the existing CS-MRMC strategy as reflected in Fig. 4. The reason is that, in the proposed FORD strategy, due to the implementation of latency-aware resource scheduling, tasks are scheduled and executed on the nearest resources with minimum latency. Therefore, there is minimum latency in the proposed FORD strategy.

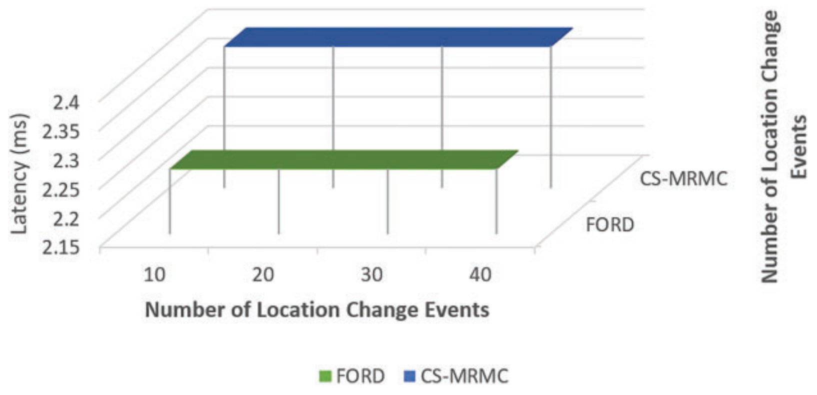


Figure 4: Latency comparison of proposed FORD strategy and CS-MRMC

5.3 *Execution Time*

The simulation results show that the proposed FORD strategy performed significantly well compared with the existing CS-MRMC strategy with respect to execution time as shown in Fig. 5. The reason is that, in the proposed FORD strategy, due to the implementation of latency-aware resource scheduling and provision of fault-tolerant mechanisms, the failed tasks are re-executed immediately upon failure, whereas, in the existing approach, the complete loop is required to be executed. Therefore, there is a minimum execution time for the proposed FORD strategy.

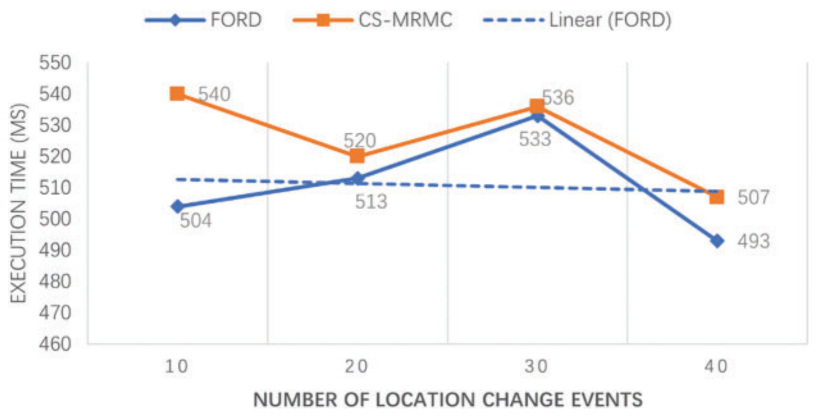


Figure 5: Execution time comparison of proposed FORD strategy and CS-MRMC

5.4 *Energy Consumption*

The simulation results show that the proposed FORD strategy significantly reduces energy consumption compared with the existing CS-MRMC strategy as given in Fig. 6. The reason is that, in the proposed FORD strategy, due to the implementation of latency-aware resource scheduling and data management, tasks are scheduled and executed on the nearest resources/nodes with minimum energy consumption. Therefore, there is minimum energy consumption for the proposed FORD strategy.

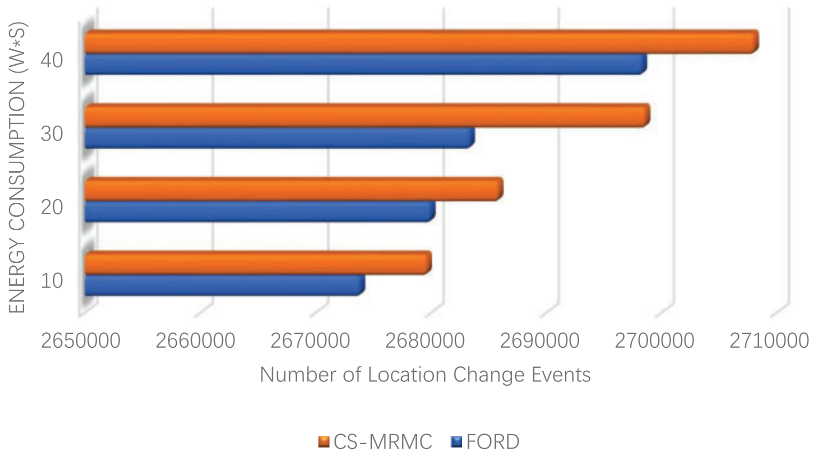


Figure 6: Energy consumption comparison of proposed FORD strategy and CS-MRMC

5.5 Discussion of Results

The simulation was implemented on iFogSim2 and validated the results through comparison with real-world benchmarks while using performance criteria such as execution cost, latency, execution time, and energy consumption. To evaluate the results, multiple simulation rounds were performed with an emphasis on low latency and fault tolerance. The outputs were analyzed closely to ensure that the resource scheduling, the execution of tasks, and the fault-tolerant mechanisms performed as expected in a fog environment. The findings were matched to real-life datasets from other comparable studies as a practical standard for validation. To further test and verify the effectiveness of the approach, additional validation in the form of sensitivity analysis of different task arrival rates, available network bandwidth, as well as node failure situations were also performed. With this process, it was also possible to check the strength of the model in other conditions. The model was also supported based on the execution cost, execution time, system throughputs, and energy utilization measures. Such evaluations prove that the proposed framework is efficient and accurate in managing the latency and fault tolerance issues efficiently in Smart City transportation systems.

By employing a decentralized architecture, the framework avoids the key issues tied to centralized systems where data can be breached. In this way, control of data is distributed between individual fog nodes, and thus the risk of failure at one point is minimized. It is believed that since access control is a component of the fog environment, it is used to control who can access or manipulate data stored in the fog nodes so that only authorized entities can access the data. The FORD framework was proposed to address three important research challenges in smart city transportation systems: latency, fault tolerance, and data management. In order to measure its efficiency, scenarios were generated with iFogSim2 and the results of the proposed strategy were compared with the CS-MRMC strategy, which is the most relevant and recent strategy in this area of research. It was found to be useful to compare these results with other state-of-the-art solutions, yet it became apparent that it was better not to make such a comparison. This was because the literature review showed that there is no other strategy that covers all three aspects of latency, fault tolerance, and data management when applied to smart city transportation systems. Thus, the CS-MRMC strategy is just used as a reference for evaluating the effectiveness of the FORD framework.

The FORD strategy is aimed at maximizing data handling from smart city transportation systems and doing this in a way that solves some important problems such as latency, fault tolerance, and efficient use of resources. To examine how to advance latency-aware resource scheduling for latency-sensitive applications in the development of a resource-scheduling framework, the FORD strategy is incorporated. This approach focuses on achieving objectives on the nearest resources conveniently, which reduces latency in data processing and decision-making. In applying fault tolerance to the latency-aware resource-scheduling framework, the FORD strategy uses reliable techniques that enable instant re-runs of failed operations without having to repeat the whole loop. This capability not only improves reliability but also improves overall execution cost, which is a major improvement over the current state-of-the-art strategy, CS-MRMC. The simulation of the FORD strategy showed that it outperforms the other approach in terms of execution cost, latency, execution time, and energy consumption. In particular, the analysis of the results of the simulation indicates that the FORD strategy is instrumental in reducing execution costs compared to the CS-MRMC strategy. Latency-aware resource scheduling and the concept of fault tolerance simplify the FORD framework's reaction to failed tasks and related costs. Concerning the latency parameter, the FORD strategy reveals remarkable enhancements compared to the CS-MRMC strategy. This improvement arises solely from the ranking of tasks according to the proximity of the required resources, thus decreasing the latency inherent to data processing. The evaluation of the FORD strategy reveals that it has a short execution time. Through latency-aware resource scheduling and fault tolerance mechanisms, the framework can easily re-execute failed tasks, and as a result, the execution time is faster than the proposed CS-MRMC strategy. Energy consumption is another area that can be considered one of the significant benefits of the FORD strategy because of the stronger performance of this strategy in comparison with the existing one. As such, by arranging task execution on the nearest resources, the FORD framework is able to minimize the energy consumption incurred in processing data as part of smart city transportation systems.

6 Conclusion and Future Work

This research presents a latency-aware resource scheduling framework for latency-sensitive applications in smart city transportation systems, termed FORD. The FORD strategy is implemented through simulations. FORD addresses three important challenges as highlighted through research questions: to reduce latency, to implement fault tolerance and to make data management efficient. The results demonstrate significant advantages of the FORD strategy over the CS-MRMC strategy across all evaluated parameters. The FORD framework exhibits superior performance in terms of execution cost, latency, execution time, and energy consumption. This superiority can be attributed to FORD's emphasis on latency-aware resource scheduling, fault tolerance mechanisms, and efficient data management techniques. The implementation of latency-aware resource scheduling ensures that tasks are executed on the nearest available resources, minimizing latency associated with data processing and decision-making within smart cities' transportation systems. Furthermore, the fault-tolerant mechanisms incorporated into the FORD framework enable prompt re-execution of failed tasks, thereby reducing overall execution time and improving system reliability. The FORD strategy optimizes energy consumption by efficiently managing task execution on the nearest resources, thereby minimizing energy expenditure associated with data processing. The FORD framework also introduces an efficient data management technique that manages large volumes of data. This technique leverages fog nodes for real-time processing while managing data storage on cloud data centres, optimizing both data transmission and processing times.

The proposed framework is evaluated through simulation; therefore, in the future, there are plans to perform real-world experiments to confirm the workability of the proposed solution. Another important assumption of the framework is the availability of fog nodes. In some cases, it may be possible that suitable fog nodes are not available for implementation. This may mean that the environment has to be further extended, which may lead to additional resource consumption. This approach to investigating fault tolerance is limited by the types of faults and their occurrences that were explored in the simulation of the smart city environment. Therefore, it is anticipated that this work will continue to further explore more fault situations; such modifications would improve the practicality of the framework in realistic environments. The simulation environment used in the proposed study is not specifically designed for smart city transportation systems but provides fog-based node implementation for transportation systems best suited for the presented study. In the future, there is also a plan to develop or use a more specific simulator to implement or extend the current study.

Acknowledgement: The authors sincerely appreciate the support from the Deanship of Scientific Research and Graduate Studies at King Khalid University, Abha, Saudi Arabia.

Funding Statement: This research was financially supported by the Deanship of Scientific Research and Graduate Studies at King Khalid University under research grant number (R.G.P.2/93/45).

Author Contributions: Ibrar Afzal: Conceptualization, Data Curation, Formal Analysis, Investigation, Software, Validation, Writing—Original Draft. Noor ul Amin: Formal Analysis, Investigation, Resources, Supervision, Validation, Visualization, Writing—Review & Editing. Zulfiqar Ahmad: Formal Analysis, Resources, Software, Supervision, Validation, Visualization, Writing—Original Draft. Abdulmohsen Algarni: Data Curation, Formal Analysis, Funding Acquisition, Project Administration, Resources, Validation, Visualization, Writing—Review & Editing. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used and/or analyzed during the current study are publicly available in iFogSim2 Repository [47]. Any query about the code or research conducted in this paper is highly appreciated and can be asked by the corresponding authors upon request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] A. A. Laghari, A. K. Jumani, and R. A. Laghari, "Review and state of art of fog computing," *Arch. Comput. Methods Eng.*, vol. 28, no. 5, pp. 3631–3643, 2021. doi: 10.1007/s11831-020-09517-y.

[2] M. Aazam and E. N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proc. Int. Conf. Adv. Inform. Netw. Appl.*, 2015, pp. 687–694. doi: 10.1109/AINA.2015.254.

[3] A. A. Sadri, A. M. Rahmani, M. Saberikamarposhti, and M. Hosseinzadeh, "Fog data management: A vision, challenges, and future directions," *J. Netw. Comput. Appl.*, vol. 174, Jan. 2021, Art. no. 102882. doi: 10.1016/j.jnca.2020.102882.

[4] K. Ahmad, M. Maabreh, M. Ghaly, K. Khan, J. Qadir and A. Al-Fuqaha, "Developing future human-centered smart cities: Critical analysis of smart city security, Data management, and Ethical challenges," *Comput. Sci. Rev.*, vol. 43, Feb. 2022, Art. no. 100452. doi: 10.1016/j.cosrev.2021.100452.

[5] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for IoT-based smart environments: A survey," *J. Cloud Comput.*, vol. 7, no. 1, pp. 1–20, 2018. doi: 10.1186/s13677-018-0123-6.

[6] G. Casella, B. Bigliardi, and E. Bottani, "The evolution of RFID technology in the logistics field: A review," *Procedia Comput. Sci.*, vol. 200, pp. 1582–1592, 2022. doi: 10.1016/j.procs.2022.01.359.

[7] M. A. Mirzaei, K. Zare, B. Mohammadi-Ivatloo, M. Marzband, and A. Anvari-Moghaddam, "Robust network-constrained energy management of a multiple energy distribution company in the presence of multi-energy conversion and storage technologies," *Sustain. Cities Soc.*, vol. 74, Nov. 2021, Art. no. 103147. doi: 10.1016/j.scs.2021.103147.

[8] P. Gupta, R. R. Kaikini, D. K. Saini, and S. Rahman, "Cost-aware resource optimization for efficient cloud application in smart cities," *J. Sens.*, vol. 2022, 2022, Art. no. 4406809. doi: 10.1155/2022/4406809.

[9] G. Huang, D. Li, L. Yu, D. Yang, and Y. Wang, "Factors affecting sustainability of smart city services in China: From the perspective of citizens' sense of gain," *Habitat Int.*, vol. 128, Oct. 2022, Art. no. 102645. doi: 10.1016/j.habitatint.2022.102645.

[10] D. Li, X. Shang, G. Huang, S. Zhou, M. Zhang and H. Feng, "Can smart city construction enhance citizens' perception of safety? A case study of Nanjing, China," *Soc Indic. Res.*, vol. 171, no. 3, pp. 937–965, Feb. 2024. doi: 10.1007/s11205-023-03304-5.

[11] C. Gomez, S. Chessa, A. Fleury, G. Roussos, and D. Preuveneers, "Internet of Things for enabling smart environments: A technology-centric perspective," *J. Ambient Intell. Smart Environ.*, vol. 11, no. 1, pp. 23–43, 2019. doi: 10.3233/AIS-180509.

[12] J. Chen, M. Xu, W. Xu, D. Li, W. Peng and H. Xu, "A flow feedback traffic prediction based on visual quantified features," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 9, pp. 10067–10075, Sep. 2023. doi: 10.1109/TITS.2023.3269794.

[13] J. Chen, Q. Wang, H. H. Cheng, W. Peng, and W. Xu, "A review of vision-based traffic semantic understanding in ITSs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 19954–19979, Nov. 2022. doi: 10.1109/TITS.2022.3182410.

[14] J. Lu and C. Osorio, "On the analytical probabilistic modeling of flow transmission across nodes in transportation networks," *Transp. Res. Rec. J. Transp. Res. Board.*, vol. 2676, no. 12, pp. 209–225, Dec. 2022. doi: 10.1177/03611981221094829.

[15] S. Zahoor and R. N. Mir, "Resource management in pervasive Internet of Things: A survey," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 33, no. 8, pp. 921–935, 2021. doi: 10.1016/j.jksuci.2018.08.014.

[16] P. Pop *et al.*, "The FORA fog computing platform for industrial IoT," *Inf. Syst.*, vol. 98, 2021, Art. no. 101727. doi: 10.1016/j.is.2021.101727.

[17] M. H. Kashani, A. M. Rahmani, and N. J. Navimipour, "Quality of service-aware approaches in fog computing," *Int. J. Commun. Syst.*, vol. 33, no. 8, pp. 1–34, 2020. doi: 10.1002/dac.4340.

[18] A. Rehman, K. Haseeb, T. Saba, and H. Kolivand, "M-SMDM: A model of security measures using green Internet of Things with cloud integrated data management for smart cities," *Environ. Technol. Innov.*, vol. 24, 2021, Art. no. 101802. doi: 10.1016/j.eti.2021.101802.

[19] J. Huang, S. Li, and Y. Chen, "Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing," *Peer-to-Peer Netw. Appl.*, vol. 13, no. 5, pp. 1776–1787, 2020. doi: 10.1007/s12083-020-00880-y.

[20] A. Alarifi, F. Abdelsamie, and M. Amoon, "A fault-tolerant aware scheduling method for fog-cloud environments," *PLoS One*, vol. 14, no. 10, pp. 1–24, 2019. doi: 10.1371/journal.pone.0223902.

[21] M. S. U. Islam, A. Kumar, and Y. C. Hu, "Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions," *J. Netw. Comput. Appl.*, vol. 180, 2021, Art. no. 103008. doi: 10.1016/j.jnca.2021.103008.

[22] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10200–10232, 2020. doi: 10.1109/JIOT.2020.2987070.

[23] A. Javed, J. Robert, K. Heljanko, and K. Främling, "IoTEF: A federated edge-cloud architecture for Fault-Tolerant IoT Applications," *J. Grid Comput.*, vol. 18, no. 1, pp. 57–80, 2020. doi: 10.1007/s10723-019-09498-8.

[24] Y. Gong, H. Yao, and A. Nallanathan, "Intelligent sensing, communication, computation, and caching for satellite-ground integrated networks," *IEEE Netw.*, vol. 38, no. 4, pp. 9–16, Jul. 2024. doi: 10.1109/MNET.2024.3413543.

[25] R. Casado-Vara, Z. Vale, J. Prieto, and J. M. Corchado, "Fault-tolerant temperature control algorithm for IoT networks in smart buildings," *Energies*, vol. 11, no. 12, pp. 1–17, 2018. doi: 10.3390/en11123430.

[26] W. -L. Liu, J. Zhong, P. Liang, J. Guo, H. Zhao and J. Zhang, "Towards explainable traffic signal control for urban networks through genetic programming," *Swarm Evol. Comput.*, vol. 88, Jul. 2024, Art. no. 101588. doi: 10.1016/j.swevo.2024.101588.

[27] Y. Wang, R. Sun, Q. Cheng, and W. Y. Ochieng, "Measurement quality control aided multisensor system for improved vehicle navigation in Urban Areas," *IEEE Trans. Ind. Electron.*, vol. 71, no. 6, pp. 6407–6417, Jun. 2024. doi: 10.1109/TIE.2023.3288188.

[28] Q. Duan *et al.*, "Optimal scheduling and management of a smart city within the safe framework," *IEEE Access*, vol. 8, pp. 161847–161861, 2020. doi: 10.1109/ACCESS.2020.3021196.

[29] W. Wen *et al.*, "Health monitoring and diagnosis for geo-distributed edge ecosystem in smart city," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 18571–18578, Nov. 2023. doi: 10.1109/JIOT.2023.3247640.

[30] R. Udayakumar *et al.*, "An integrated deep learning and edge computing framework for intelligent energy management in IoT-based smart cities," in *2023 Int. Conf. Technol. Eng. Appl. Sustainable Dev. (ICTEASD)*, IEEE, Nov. 2023, pp. 32–38. doi: 10.1109/ICTEASD57136.2023.10585232.

[31] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, 2018, Art. no. 9. doi: 10.1145/3186592.

[32] H. Wadhwa and R. Aron, "TRAM: Technique for resource allocation and management in fog computing environment," *J. Supercomput.*, vol. 78, no. 1, pp. 667–690, 2022. doi: 10.1007/s11227-021-03885-3.

[33] Y. Xiao and M. Krunz, "AdaptiveFog: A modelling and optimization framework for fog computing in intelligent transportation systems," *IEEE Trans. Mob. Comput.*, Dec. 2021. doi: 10.1109/TMC.2021.3080397.

[34] H. A. Khattak, H. Farman, B. Jan, and I. Ud Din, "Toward integrating vehicular clouds with IoT for smart city services," *IEEE Netw.*, vol. 33, no. 2, pp. 65–71, Mar. 2019. doi: 10.1109/MNET.2019.1800236.

[35] A. Lakhan, M. A. Dootio, T. M. Groenli, A. H. Sodhro, and M. S. Khokhar, "Multi-layer latency aware workload assignment of E-transport IoT applications in mobile sensors cloudlet cloud networks," *Electron*, vol. 10, no. 14, Jul. 2021, Art. no. 1719. doi: 10.3390/electronics10141719.

[36] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Towards fault tolerant fog computing for IoT-based smart city applications," in *2019 IEEE 9th Annu. Comput. Commun. Work. Conf. CCWC 2019*, 2019, pp. 752–757. doi: 10.1109/CCWC.2019.8666447.

[37] W. Saeed, Z. Ahmad, A. I. Jehangiri, N. Mohamed, and A. I. Umar, "A fault tolerant data management scheme for healthcare Internet of Things in fog computing," *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 1, pp. 35–57, 2021.

[38] O. R. Merad-Boudia and S. M. Senouci, "An efficient and secure multidimensional data aggregation for fog-computing-based smart grid," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6143–6153, 2021. doi: 10.1109/JIOT.2020.3040982.

[39] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "A novel architecture for efficient fog to cloud data management in smart cities," in *Proc. Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 2622–2623. doi: 10.1109/ICDCS.2017.202.

[40] C. Chilipirea, A. C. Petre, L. M. Groza, C. Dobre, and F. Pop, "An integrated architecture for future studies in data processing for smart cities," *Microprocess. Microsyst.*, vol. 52, pp. 335–342, Jul. 2017. doi: 10.1016/j.micpro.2017.03.004.

[41] D. Puiu, S. Bischof, B. Serbanescu, S. Nechifor, J. Parreira and H. Schreiner, "A public transportation journey planner enabled by IoT data analytics," in *Proc. 2017 20th Conf. Innov. Clouds, Internet Netw., ICIN 2017*, 2017, pp. 355–359. doi: 10.1109/ICIN.2017.7899440.

[42] S. Chavhan, D. Gupta, B. N. Chandana, A. Khanna, and J. J. P. C. Rodrigues, "IoT-based context-aware intelligent public transport system in a metropolitan area," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6023–6034, Jul. 2020. doi: 10.1109/JIOT.2019.2955102.

[43] Z. Sun *et al.*, "A resource allocation scheme for edge computing network in smart city based on attention mechanism," *ACM Trans. Sens. Netw.*, Mar. 2024. doi: 10.1145/3650031.

[44] I. Rafiq, A. Mahmood, S. Razzaq, S. H. M. Jafri, and I. Aziz, "IoT applications and challenges in smart cities and services," *J. Eng.*, vol. 2023, no. 4, Apr. 2023, Art. no. e12262. doi: 10.1049/tje2.12262.

[45] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Softw.: Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017. doi: 10.1002/spe.2509.

[46] S. Mustafa, B. Nazir, A. Hayat, A. Ur Rehman Khan, and S. A. Madani, "Resource management in cloud computing: Taxonomy, prospects, and challenges," *Comput. Electr. Eng.*, vol. 47, pp. 186–203, 2015. doi: 10.1016/j.compeleceng.2015.07.021.

[47] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *J. Syst. Softw.*, vol. 190, Sep. 2022, Art. no. 111351. doi: 10.1016/j.jss.2022.111351.