Computers, Materials & Continua
DOI: 10.32604/cmc.2023.029913
Article

An Optimized Offloaded Task Execution for Smart Cities Applications

Ahmad Naseem Alvi[1], Muhammad Awais Javed[1,*], Mozaherul Hoque Abul Hasanat[2], Muhammad Badruddin Khan[2], Abdul Khader Jilani Saudagar[2] and Mohammed Alkhathami[2]

[1]Department of Electrical and Computer Engineering, COMSATS University Islamabad, 45550, Pakistan
[2]Information Systems Department, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 11432, Saudi Arabia
*Corresponding Author: Muhammad Awais Javed. Email: awais.javed@comsats.edu.pk
Received: 15 March 2022; Accepted: 07 May 2022

Abstract: Wireless nodes are one of the main components in different applications that are offered in a smart city. These wireless nodes are responsible to execute multiple tasks with different priority levels. As the wireless nodes have limited processing capacity, they offload their tasks to cloud servers if the number of tasks exceeds their task processing capacity. Executing these tasks from remotely placed cloud servers causes a significant delay which is not required in sensitive task applications. This execution delay is reduced by placing fog computing nodes near these application nodes. A fog node has limited processing capacity and is sometimes unable to execute all the requested tasks. In this work, an optimal task offloading scheme that comprises two algorithms is proposed for the fog nodes to optimally execute the time-sensitive offloaded tasks. The first algorithm describes the task processing criteria for local computation of tasks at the fog nodes and remote computation at the cloud server. The second algorithm allows fog nodes to optimally scrutinize the most sensitive tasks within their task capacity. The results show that the proposed task execution scheme significantly reduces the execution time and most of the time-sensitive tasks are executed.

Keywords: Smart cities; fog computing; task offloading; knapsack

1 Introduction

Smart cities are emerging rapidly because they provide ease in the lifestyle of citizens. By providing intelligent transportation systems, flawless security, and water, waste, and energy management using Information and Communication Technology (ICT), smart cities improve the quality of life [1–6]. In addition, smart cities promote the economic growth of the city by optimizing its different functions through data analysis.

Smart cities offer many applications and services to benefit humanity. In transportation, in addition to moving toward electric vehicles to make it environmental pollution, real-time information from users avoids traffic congestion by disseminating the same information to other vehicles. Energy consumption and water consumption can be smartly managed with the help of smart grids and water

monitoring sensors. Safety and security are the main concern in all societies, use of security cameras with face recognition and smoke and fire sensors can provide timely responses to avoid any mishap in the smart cities [7–12].

Smart cities are based on the Internet of Things (IoT) that comprise ubiquitously deployed different types of sensor devices. The data gathered from these sensor nodes are smartly analyzed to get an insight into the different services and operations of the smart city. The analyzed data is communicated to the decision-makers for proper action. ICT framework helps decision-makers to improve their decision by collecting real-time data from different devices and machines. In addition, citizens can directly connect with the ecosystem of the smart city through mobile devices and building infrastructures. This helps in improving sustainability and making cost-effective decisions with reduced traffic congestion, improved security, and reduced air pollution [13–17].

Fog computing is an architecture that comprises one or more data centers placed at the edge of user networks and not routed over the Internet backbone. Being a part of the same IoT network gives provision to different sensor devices to send their data at a faster speed for quick processing [18–20]. The fog computing paradigm does not stick to one architecture only, however, it pushes data towards multiple edge nodes instead of processing a single node. Data from different sensor devices or IoT nodes are processed by their nearest edge node. These edge nodes have different storage capacities as well as varying processing capabilities. As an example, wireless sensor nodes have low processing power whereas smartphones laptops, palm drives have higher storage and computational capacity. Different smart city applications are shown in Fig. 1.

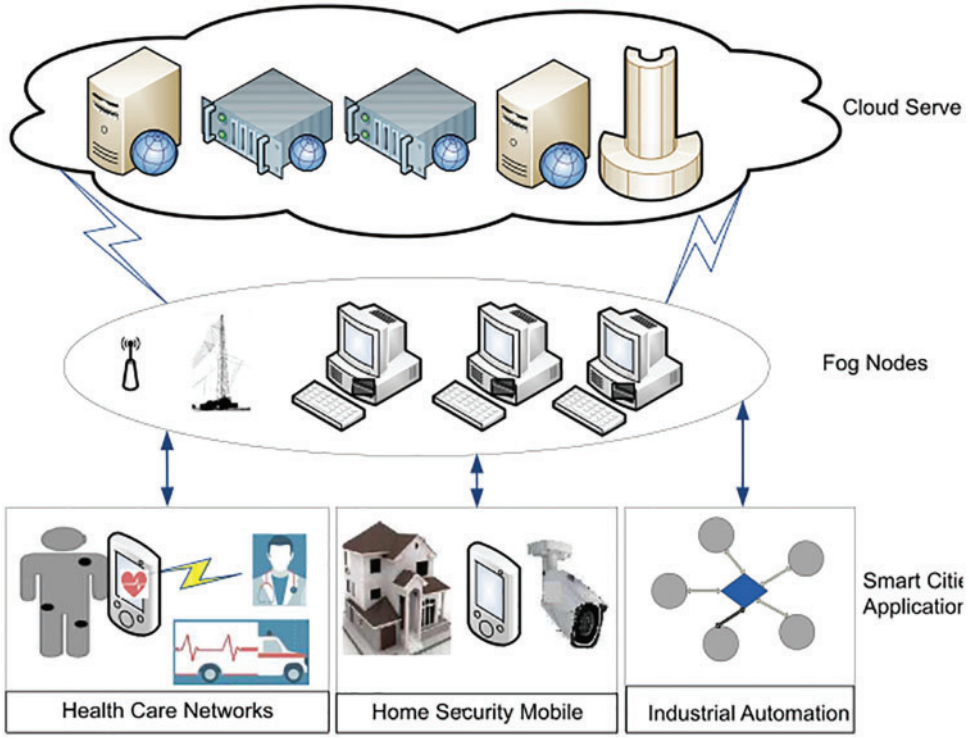

Figure 1: Fog and cloud computing in smart cities applications

Fog computing performs similar tasks as designated for cloud computing. However, it is preferred over cloud computing due to being closer to the end devices. Fog computing by doing intelligent sensing reduces the load on cloud capacity by discarding useless data. In addition, data synchronization techniques such as push schemes, pull schemes, and push-pull schemes are used for data collection from lower-end fog nodes to the higher-end fog nodes for manageable processing. All these schemes follow communication protocols according to the fog computing scenario.

One of the prime benefits of fog computing is less delay as compared to the cloud because moving data from sensor nodes to the cloud computing backbone requires time and real-time action is not possible. Several time-sensitive applications in smart cities require real-time response such as a security scenario in which a malicious user is detected using a motion sensor or a camera. Similarly, in health care, if the vital signs of a patient are disturbed, then real-time attention with a timely response is required to save the patient. Some IoT-based machines require online physical interaction to troubleshoot the problem to avoid any major loss. In all such scenarios, fog computing is preferred over cloud computing.

As IoT applications in smart cities require the dissemination of a huge amount of data from the sensors to the computing servers, fog nodes are ideally suited for the timely processing of this data. Processing time-sensitive applications with efficient task scheduling will enhance the performance of the smart city application. In contrast to the cloud, the fog not only performs latency-sensitive applications at the edge of the network but also performs latency-tolerant tasks efficiently at powerful computing nodes at the intermediate of the network. At the top of the fog, cloud computing with data centers is used for in-depth analysis of smart city data and making decisions over a longer period.

With the increase in the number of smart city applications, there is a large number of offloaded tasks requests by different IoT nodes. As fog nodes have limited task processing capabilities and cannot entertain all offloaded task requests. An optimal task processing is required to intelligently execute those offloaded tasks that have higher priority as compared to low priority tasks.

In this work, an optimal Offloaded Task Execution procedure for Smart Cities (OTE_{SC}) is proposed. The proposed scheme comprises two algorithms to timely execute the offloaded tasks for multiple smart city applications. Salient features of our algorithms are shown below:

- A task processing mechanism for the fog computing node is proposed to decide whether to forward the tasks to the cloud or processed them locally.
- If the number of tasks to be performed by fog nodes is more than their computing capacity, then OTE_{SC} allows fog nodes to optimally scrutinize tasks for execution by applying a 0/1 knapsack. The decision is based on their sensitivity level as determined in the utility function. Tasks that cannot be processed by the fog nodes are forwarded to the cloud for their execution.

The rest of the paper is organized as follows: Section 2 describes different research works regarding task offloading and fog computing. The system model and the proposed scheme are elaborated in Sections 3 and 4 respectively. The performance comparison of OTE_{SC} with other schemes is discussed in Sections 5 and 6 concludes the paper.

2 Related Works

There is a lot of ongoing research in the area of fog computing because it offers reduced delay as compared to the cloud computing servers. The utility of fog computing is under hot research in different application scenarios of smart cities.

In [21], a conventional model of task offloading and allocating resources is proposed for smart cities applications. The proposed scheme evaluates the performance of various units to accommodate the use of artificial intelligence in the sixth-generation wireless communication era.

A task offloading optimization technique by making the trade-off between consumed energy and response time of nodes is proposed in [22]. Based on these two trade-offs, the authors proposed a bi-objective optimization algorithm by modeling these two sub-objectives.

In [23], offloaded task processing of a large number of smart city users is proposed by considering their energy and time sensitivity with privacy preservation. The authors claimed that their proposed scheme effectively improves the resource utilization and energy consumption of the smart city system. Reference [24] focused on the limited battery and efficiency of Unmanned Aerial Vehicles (UAVs), which are used to collect the sensing data from different sensing nodes in a smart city. The authors proposed an offloaded architecture for UAVs by developing an offloaded model to select the optimal vehicles in such a way that their efficacy may be improved. The authors claimed that their proposed scheme significantly improved the utilities of UAVs and vehicles in a smart city.

A solution by considering the limited battery power with computational and communications capacity of smart mobile devices for a smart city is proposed in [25]. The authors developed a framework by introducing a mobile cloud computing model. The model illustrates the data and operational flows of different applications in the smart city by allowing load balancing among different environmental entities specified in the mobility and traffic control domains.

As compared to the previous work, our proposed algorithm focuses on time-sensitive application tasks for smart cities applications and utilizes a Knapsack-based algorithm to maximize the computational speed of high priority tasks while meeting the capacity requirements of the fog nodes.

3 System Model

In this work, we consider three smart city applications as health care, IoT-based industrial automation, and home security applications. All these applications have been divided into three groups of tasks, such as emergency tasks, alert tasks, and normal routine tasks with their severity levels as high, medium, and low respectively. The sizes of these tasks vary and even task sizes of the same severity levels in the same application are not the same.

Nodes offload their desired tasks for onward processing. The offloaded tasks are supposed to be processed either by remotely placed cloud servers or nearby computing servers called fog nodes. Application nodes have direct access to these fog nodes with better data rates. These fog nodes are backwardly connected with cloud servers through the internet backbone. The tasks, that are required to be processed by the cloud servers offer a relatively low data rate as compared to fog nodes.

In this work, we consider N fog computing nodes in sparsely placed different smart cities application networks. Each N_i fog node is connected with K smart city application networks. Each of the smart city application networks K_j has three different types of tasks and are categorized as t_1, t_2, and t_3 with low to high priority respectively. T is the total number of tasks that are required to be executed and are calculated as:

$$T = \sum_{i=1}^{V} \sum_{j=1}^{K} N_i t_j \tag{1}$$

The downloaded data rate in processing tasks of a node from fog node (DR_1) is computed as:

$$DR_1 = log_2(1 + \gamma_{n,f}) \tag{2}$$

Here $\gamma_{n,f}$ is signal to noise ratio between the application node and fog computing node.

Downloading data rate of same processing task from cloud server (DR_2) is calculated as:

$$DR_2 = log_2(1 + \gamma_{n,c}) \tag{3}$$

Here $\gamma_{n,c}$ is signal to noise ratio between the application node and cloud server. We assume that there are multiple channels available in each fog node to transmit all executed tasks simultaneously without considering any queuing delay.

4 Proposed Task Execution Policy

In this work, we propose an optimal Offloaded Task Execution procedure for Smart Cities (OTE_{SC}). The proposed scheme allows fog computing nodes to execute those tasks that are offloaded by different nodes and are placed in the close vicinity of different applications of a smart city. Fog nodes are located nearby these smart city applications to minimize the task delay. Each fog node has limited simultaneous processing capability. However, the cloud has huge processing and storing capability. The offloaded tasks are supposed to be processed by fog nodes. In case, offloaded tasks are more than the processing capacity of its fog node, then they are forwarded to a centralized cloud server for task execution. The offloaded tasks have different priority levels with strict delay constraints. It is therefore required that sensitive offloaded tasks should be processed quickly and preferably should be executed by the fog node as shown in Algorithm 1. As offloaded tasks are different in size; optimal decision-making is required to perform those tasks within the task processing capacity of the fog node. The proposed OTE_{SC} optimally scrutinizes offloaded tasks according to their task priority and sizes within the task processing limit of the fog node by applying task selection criteria and a 0/1 knapsack optimizing algorithm.

Algorithm 1: Task Processing Criteria

$y_i \leftarrow$ *Current offloaded task*
$i \leftarrow$ *current task offloading node ID*
$F_C \leftarrow$ *Task execution capacity of fog node*
$T \leftarrow$ *Task size*
$TS \leftarrow$ *Maximum offloaded task size*
$x_i \leftarrow$ *Current task offloading node*
$X \leftarrow$ *Max. no. of tasks requesting nodes*
$y_i \leftarrow$ *content size requested by* i^{th} *node*
Task execution policy
for $x = 1$ *to* X **do**
  for $y_x = 1$ *to* T **do**
    $TS = \sum_{x=1}^{X} \sum_{y_1}^{Y} T[N_x(t_y)]$
    If $TS \leq F_C$ *Execute all offloaded tasks at fog node*
    Else
    Apply 0/1 knapsack to scrutinize tasks for execution by fog node
    EndIf
  end
end

4.1 Task Selection Criteria

Fog nodes in different places of a smart city are receiving offloaded tasks at regular intervals. Each fog node has a limited task processing capacity and keeps on processing offloaded tasks after regular time intervals.

If a fog node has CAP_{TE} task execution capacity and X nodes intend to offload their different priorities tasks at a specific time. If offloaded task size of one of the nodes X_i is OT_i then the total offloaded task sizes Tot by X nodes is calculated as:

$$T_{ot} = \sum_{i=1}^{X} OT_i \tag{4}$$

If

$$T_{ot} \leq CAP_{TE} \tag{5}$$

then, the fog node will execute all offloaded task requests itself, otherwise, it needs to forward some of the tasks to the cloud servers, that are beyond its task execution capacity.

The proposed OTE_{SC} offers the optimal selection of offloaded tasks to be performed by fog nodes with respect to their priority and task sizes. The complete task selection criteria are shown in Algorithm 2.

4.2 0l1 Knapsack for Task Scheduling

Fog nodes have limited execution capacity and if requested tasks are more than their computing capacity then it needs to scrutinize those offloaded tasks that have more priority.

Different applications of smart cities have varying priorities of tasks with adaptive task sizes. Optimal scrutiny of different sized task requests in fixed data execution capacity of fog node is solved by 0/1 knapsack problem.

Knapsack algorithm can be applied to scrutinize these offloaded tasks according to their priorities. knapsack picks the most valuable items from the given list up to their maximum capacity. Our designed problem can be mapped with the 0/1 knapsack problem in the following way. The knapsack carrying capacity is mapped with a maximum task processing capacity of the fog node. The weight of an item is mapped with offloaded task sizes. The value of the item is mapped with task priority.

Suppose there are T tasks, that have been offloaded by N smart city nodes, and these tasks are categorized as high, medium, and low priority tasks. If CAP_{TE} is the maximum task execution capacity of a fog node at a specific time, then 0/1 knapsack optimally scrutinized offloaded task for processing with the following constraints: The task processing time of all the scrutinized offloaded tasks should not be more than the task processing limit of the fog node and is expressed as:

$$\sum_{i=1}^{N} \sum_{j=1}^{T} k_i t_j \leq CAP_{TE} \tag{6}$$

The second constraint is that the task priorities of all the scrutinized tasks must be maximum.

$$max \sum_{a=1}^{K} P_K \tag{7}$$

Algorithm 2: Task Selection Criteria

```
y ← Current task size
Y ← Max. task execution capacity of fog node
i ← Task ID
n ← Max. no. of tasks
X[i, y] ←
   Cell value of table with i^th task and y processing
y_t ← task size requested by i^th task
filling of knapsack table:
for y = 0 to Y do
    X[0, y] = 0
    // Initialize 1st row to 0's
end
for i = 1 to v do
    X[i, 0] = 0
    // Initialize 1st column to 0's
end
for i = 1 to v do
    for y = 0 to Y do
        If y_t ≤ y  If y_t + X[i − 1, y − y_t] > X[i − 1, y]
        X[i, y] = y_t + X[i − 1, y − y_t]
        Else
        X[i, y] = X[i − 1, y]
        EndIf
        Else
        X[i, y] = X[i − 1, y]
        EndIf
    end
     Initialize i and w:
    v ← i
    Y ← m
end
optimal task selection:
while i > 1 and y > 1 do
    If B[i, y] > B[i − 1, y]
    i^th content is included in optimal solution
    i = i − 1
    y = y − y_t
    Else
    i = i − 1
    EndIf
end
```

5 Performance Evaluation

In this section, the comparative performance analysis of our proposed OTE_{SC} is analyzed in different prospects. Different nodes of smart cities applications offload their different priority tasks to their nearby fog node. Task offloading nodes are randomly chosen to forward one to three different priorities of tasks. The performance of our proposed OTE_{SC} is compared with the following two task offloading schemes. In the first scheme, each fog computing node processes offloaded tasks randomly within its task processing capacity and forwards the unprocessed tasks to cloud servers. In the second scheme, the fog node computes offloaded tasks on First Come First Serve (FCFS) basis up to its processing capacity and forwards the rest to a cloud server for task execution.

To compare our proposed scheme with the other two schemes, a simulation environment is created to obtain results in different prospects for fair comparative analysis. A MATLAB simulator is used to simulate the comparative analysis of our proposed scheme with the other two schemes by creating a simulator environment as discussed in Section 3. In this simulation, three different task sensitivity levels from three different smart cities applications are taken. The tasks sizes with different sensitivity levels are ranged from 50 to 150 bytes for most sensitive tasks, 120 to 250 bytes for a medium level of sensitive tasks, and 150 to 2500 bytes for routine tasks with the least priority level.

The offloaded tasks processed from fog computing nodes are downloaded at the data rate of 10 Mbps. However, downloading data rate from the cloud is taken as 4 Mbps. The simultaneous task processing capability of a fog node is taken as 500 MBytes, whereas the cloud has unlimited task processing capacity. As both fog and cloud execute tasks in a quick span, the task processing time is ignored, and only downloading time of executed tasks is calculated.

A Monte Carlo-based simulation is performed, and results are obtained against 10^3 iterations for a fair comparative analysis of OTE_{SC}. Salient simulation parameters are listed in Tab. 1.

Table 1: Simulation parameters

Parameter	Value
Fog node coverage area	2000 m
Distance between fog node and offloaded task requesting nodes	50–2000
Number of simultaneous offloaded tasks for each fog node	100–600
Number of fog nodes	2–10
Download data rate from cloud	4 Mbps
Emergency tasks sizes of different applications	50–150
Medium sensitivity level tasks of different applications	120–250
Routine tasks with least sensitivity	150–2500

The performance of our proposed OTE_{SC} is analyzed with the other two schemes. The results in different prospects are analyzed for varying numbers of tasks of different sensitivity levels and a varying number of fog nodes.

Fog computing nodes are placed in different locations of the smart city to provide easy access to its different application nodes. This provides quick execution of offloaded tasks received from smart city nodes as compared to a remotely placed cloud server. A comparative analysis of downloading the executed tasks from fog nodes and cloud servers is shown in Fig. 2. The results clearly show that downloading time of executed tasks from fog nodes is much smaller than the cloud servers.

The comparative results of total offloaded requested tasks performed by the fog node by applying OTE_{SC} and with the other two schemes are shown in Fig. 3. The figure comprises three sub-figures representing three different priority levels tasks. The results show that the proposed scheme allows the fog node to process twice the number of high-priority tasks as compared to the other two schemes. At the same time, the proposed scheme executes slightly more medium priority tasks as compared to its competitors because it still has some capacity to execute some medium level tasks after executing the majority of the high priority tasks. However, for low-priority tasks, OTE_{SC} is unable to execute any tasks because the task execution capacity of the fog node is fulfilled.

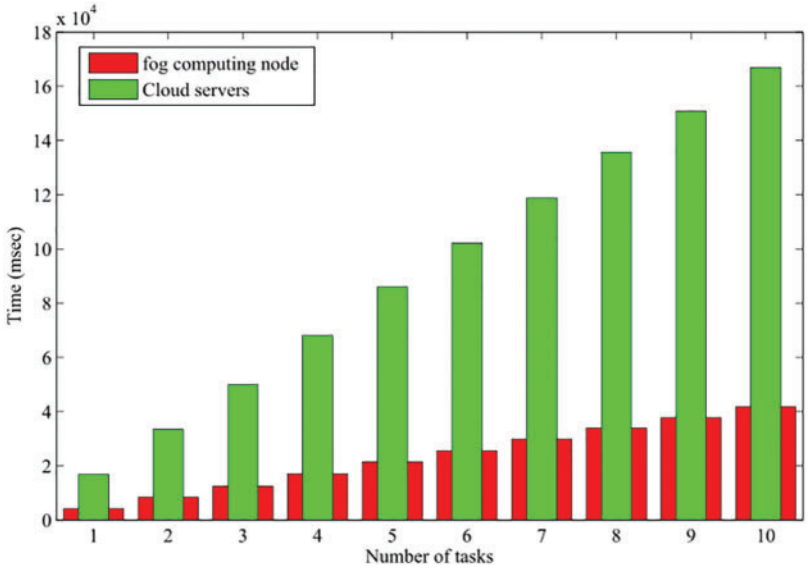

Figure 2: Task execution time in fog and cloud servers for varying number of RSUs

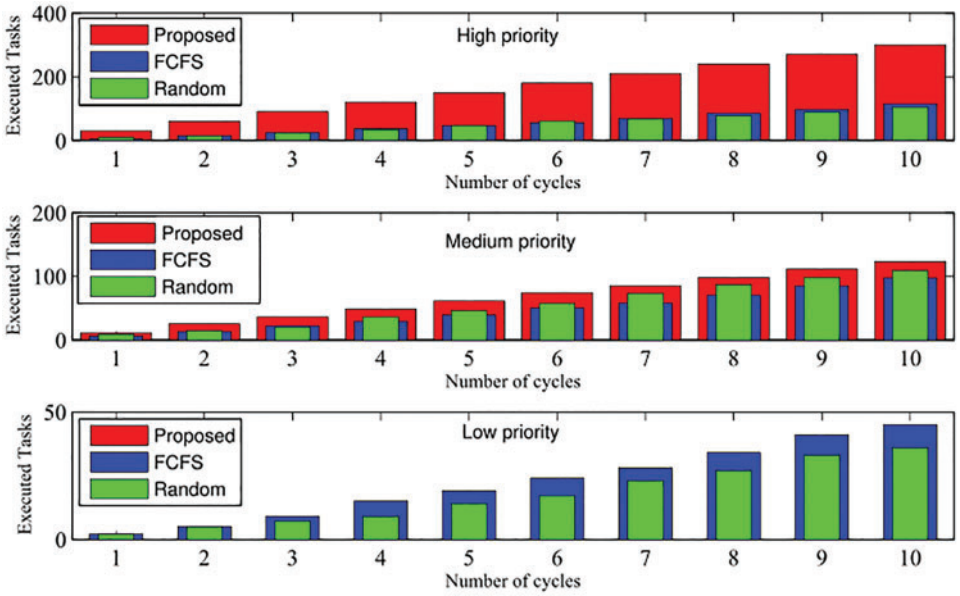

Figure 3: Number of tasks executed by fog nodes in different processing cycles

For a better comparative analysis, the percentage of executed tasks for all three different priority tasks is calculated in Fig. 4. It is evident from the results that, task execution in OTE_{SC} for high priority tasks is 100% whereas the percentage of task execution in the other two schemes is less than 50%. For medium priority tasks, the task execution percentage is more in the proposed scheme as compared to the other two schemes. However, the proposed scheme could not execute low-priority tasks because the fog node does not have fog processing capacity anymore.

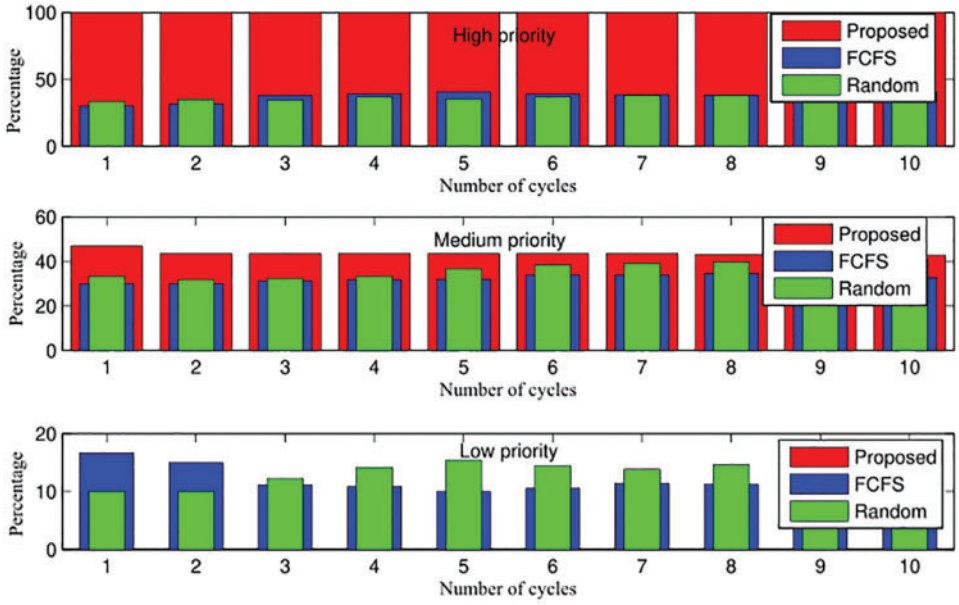

Figure 4: Percentage of tasks executed by fog nodes in different processing cycles

Fig. 5 calculates the time required to execute three different priority tasks for each scheme. The results show that the accumulated time required to execute the high-priority tasks in the proposed scheme is significantly less than in the other two schemes. Because OTE_{SC} scrutinizes high priority tasks to be executed by fog node and not a single high priority task is forwarded to the cloud server for execution. On the other hand, the other two schemes allow the fog node to execute some of its requested tasks and the rest are forwarded to a cloud server for their execution. That's why task processing time in the proposed scheme is significantly less. Task execution time for medium priority tasks in OTE_{SC} is still less than the other two schemes because OTE_{SC} executes most of the medium priority tasks at the fog node as compared to the other two schemes and fewer medium priority tasks are forwarded to cloud servers for task execution as compared to other two schemes. However, tasks processing time in OTE_{SC} is significantly large than the other two schemes for low priority tasks execution because the fog node has already reached its task execution capacity and all the low priority tasks are executed by cloud servers and consequently consume more time as compared to other two schemes.

Fig. 6 shows a performance analysis of the proposed scheme with the other two schemes for varying number of tasks requests received at the fog computing node in each interval of time. The results are obtained by computing the total executed tasks for all performance cycles when each type of the number of requested tasks is increased gradually in each performance monitoring cycle. The results show that the proposed OTE_{SC} allows the fog node to execute all of the high-priority tasks throughout. In comparison, the other two schemes do not execute all tasks and the difference in task execution for high priority tasks increases with the increase in task requests. In addition, the medium priority tasks are also processed more than the other two schemes. However, for low-priority tasks, the performance of OTE_{SC} is compromised because of limited fog computing capability. The difference in task execution increases with the increase in task requests because the proposed scheme optimally scrutinizes higher priority tasks as compared to low priority tasks.

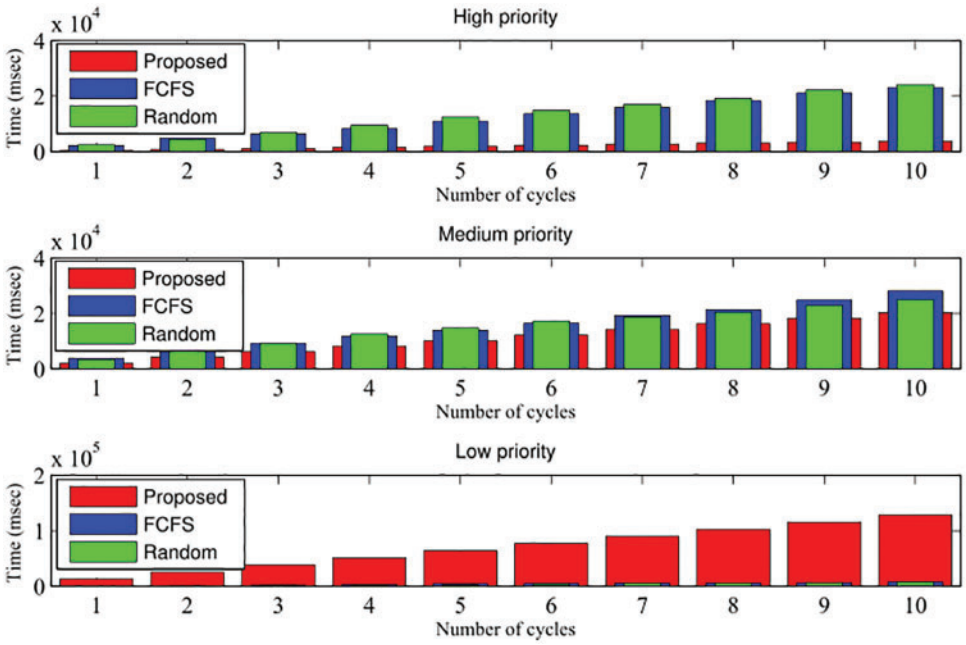

Figure 5: Number of tasks executed by fog nodes for varying number of requested tasks

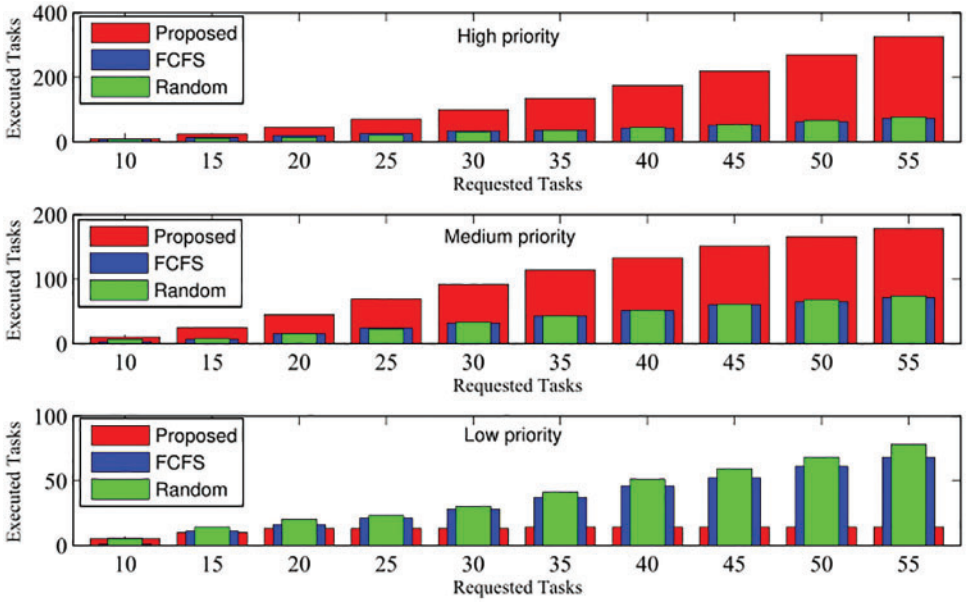

Figure 6: Percentage of tasks executed by fog nodes for varying number of requested tasks

The results shown in Fig. 7 provide a clearer picture of the executed tasks by showing the percentage of tasks executed by fog nodes. It is evident from the results that OTE_{SC} helps fog nodes to execute 100% high priority tasks as compared to 25% and 22% tasks execution by FCFS and randomly chosen tasks. In addition, OTE_{SC} executes 100% medium priority tasks until each type of requested task exceeds 25. When requested tasks are more than 25 tasks each, then task execution of medium

priority tasks gradually decreases. However, it still performs better than the other two schemes. For low priority tasks, the other two schemes execute more tasks as compared to the proposed scheme, and this difference increases with the increase in task requests.

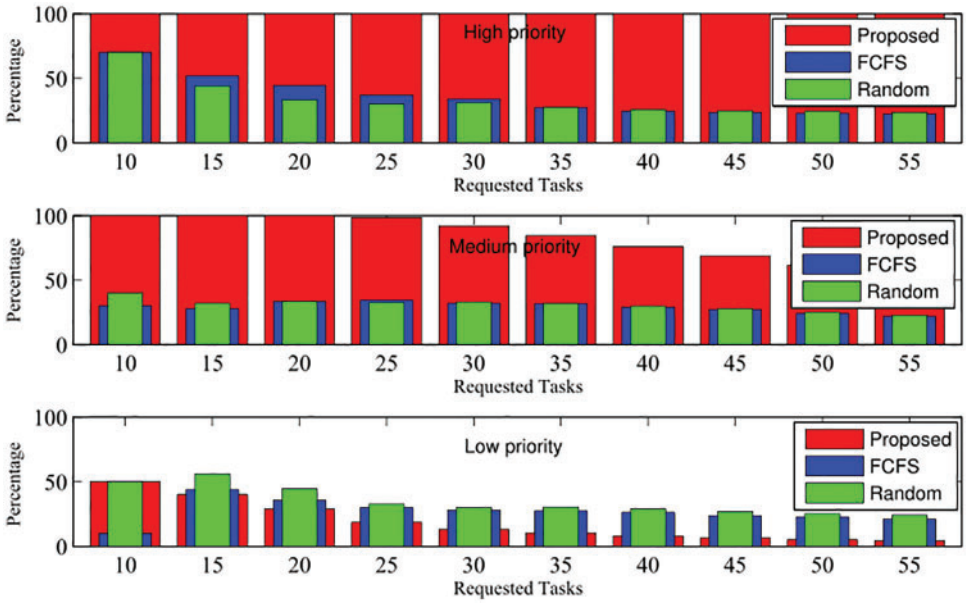

Figure 7: Accumulated processing task time for different processing cycles

6 Conclusion

In smart cities applications, fog computing nodes are preferred over cloud computing servers to minimize the execution delay of offloaded tasks. In this paper, an optimal offloaded task execution procedure for smart cities OTE_{SC} is proposed. It comprises two algorithms, that are proposed for fog computing nodes to efficiently execute the offloaded tasks. Initially, the offloaded tasks are computed to determine the offloaded tasks within their processing capacity. In case, offloaded tasks are more than the fog node capacity, then it scrutinizes offloaded tasks to be processed by the fog node by applying the 0/1 knapsack algorithm. To evaluate its performance, three different priority tasks for three smart city applications are taken. The designed algorithms efficiently scrutinize the most sensitive tasks from different applications of smart cities and the rest are forwarded to cloud servers for processing. The results are compared with randomly processed tasks and FCFS techniques. It is evident from the results that the proposed scheme allows fog nodes to execute 100% highest priority tasks with an average task processing time of less than 14 ms. In addition, the proposed scheme allows fog nodes to process about 48% of medium priority tasks and the rest are forwarded to the cloud server for their execution. In the future, we will work on the challenges such as load balancing and fair task offloading in fog computing networks.

Acknowledgement: The authors extend their appreciation to the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University for funding this work through Research Group no. RG-21-07-06.

Funding Statement: The authors extend their appreciation to the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University for funding this work through Research Group no. RG-21-07-06.

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

References

[1] S. Bijjahalli and R. Sabatini, "A high-integrity and low-cost navigation system for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 356–369, 2021.

[2] J. Kaur, S. Ahmed, Y. Kumar, A. Alaboudi, N. Z. Jhanjhi *et al.,* "Packet optimization of software defined network using lion optimization," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 2617–2633, 2021.

[3] S. Goyal, S. Bhushan, Y. Kumar, A. H. S. Rana, M. R. Bhutta *et al.,* "An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm," *Sensors*, vol. 21, no. 5:1583, pp. 1–20, 2021.

[4] M. Rahim, M. A. Javed, A. N. Alvi and M. Imran, "An efficient caching policy for content retrieval in autonomous connected vehicles," *Transportation Research Part A: Policy and Practice*, vol. 140, no. 3, pp. 142–152, 2020.

[5] S. Rani, D. Koundal, Kavita, M. F. Ijaz, M. Elhoseny *et al.,* "An optimized framework for WSN routing in the context of industry 4.0," *Sensors*, vol. 21, no. 19:6474, pp. 1–15, 2021.

[6] D. Gupta, S. Rani, S. H. Ahmed, S. Verma, M. F. Ijaz *et al.,* "Edge caching based on collaborative filtering for heterogeneous ICN-IoT applications," *Sensors*, vol. 21, no. 16:5491, pp. 1–17, 2021.

[7] M. A. Javed, S. Zeadally and E. B. Hamida, "Data analytics for cooperative intelligent transport systems," *Vehicular Communications*, vol. 15, no. 7, pp. 63–72, 2019.

[8] A. Lakhan, M. A. Mohammed, O. I. Obaid, C. Chakraborty, K. H. Abdulkareem *et al.,* "Efficient deep-reinforcement learning aware resource allocation in SDN-enabled fog paradigm," *Automated Software Engineering*, vol. 29, no. 20, pp. 1–25, 2022.

[9] S. Zeadally, M. A. Javed and E. B. Hamida, "Vehicular communications for its: Standardization and challenges," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 11–17, 2020.

[10] A. Lakhan, M. A. Mohammed, D. A. Ibrahim and K. H. Abdulkareem, "Bio-inspired robotics enabled schemes in blockchain-fog-cloud assisted IoMT environment," *Journal of King Saud University-Computer and Information Sciences*, 2022.

[11] M. A. Javed, M. Z. Khan, U. Zafar, M. F. Siddiqui, R. Badar *et al.,* "ODPV: An efficient protocol to mitigate data integrity attacks in intelligent transport systems," *IEEE Access*, vol. 8, pp. 114733–114740, 2020.

[12] J. Lee, S. Kwak, S. Park and S. Park, "Cluster-based stable BSM dissemination system for safe autonomous platooning," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 321–338, 2022.

[13] A. A. Mutlag, M. K. A. Ghani, M. A. Mohammed, A. Lakhan, O. Mohd *et al.,* "Multi-agent systems in fog-cloud computing for critical healthcare task management model (CHTM) used for ECG monitoring," *Sensors*, vol. 21, no. 20:6923, pp. 1–17, 2021.

[14] A. Lakhan, M. A. Mohammed, A. N. Rashid, S. Kadry, T. Panityakul *et al.,* "Smart-contract aware ethereum and client-fog-cloud healthcare system," *Sensors*, vol. 21, no. 12:4093, pp. 1–21, 2021.

[15] A. Nassar and Y. Yilmaz, "Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 222–235, 2022.

[16] A. Lakhan, Q. Mastoi, M. Elhoseny, M. S. Memon and M. A. Mohammed, "Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using IoT assisted mobile fog cloud," *Enterprise Information Systems*, pp. 1–23, 2021.

[17] M. A. Javed and S. Zeadally, "AI-empowered content caching in vehicular edge computing: Opportunities and challenges," *IEEE Network*, vol. 35, no. 3, pp. 109–115, 2021.

[18] S. K. Pande, S. K. Panda, S. Das, K. S. Sahoo, A. K. Luhach *et al.,* "A resource management algorithm for virtual machine migration in vehicular cloud computing," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 2647–2663, 2021.
[19] F. Jameel, M. A. Javed, S. Zeadally and R. Jäntti, "Efficient mining cluster selection for blockchain-based cellular V2X communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4064–4072, 2021.
[20] S. Gyawali, S. Xu, Y. Qian and R. Q. Hu, "Challenges and solutions for cellular based V2X communications," *IEEE Communications Surveys &Tutorials*, vol. 23, no. 1, pp. 222–255, 2021.
[21] S. Jamil, M. Arif Khan and S. Rehman, "Intelligent task off-loading and resource allocation for 6G smart city environment," in *Proc. IEEE Conf. on Local Computer Networks (LCN)*, Sydney, Australia, pp. 441–444, 2020.
[22] S. Li, "A task offloading optimization strategy in MEC-based smart cities," *Internet Technology Letters*, vol. 4, no. 158, pp. 1–5, 2020.
[23] B. Zhao, K. Peng, H. Zhang and X. Xu, "Energy-and time-efficient tasks offloading and dynamic resource allocation in smart city," in *Proc. Int. Conf. on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, Melbourne, Australia, pp. 705–712, 2021.
[24] M. Dai, Z. Su, Q. Xu and N. Zhang, "Vehicle assisted computing offloading for unmanned aerial vehicles in smart city," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1932–1944, 2021.
[25] D. Mazza, D. Tarchi and G. E. Corazza, "A unified urban mobile cloud computing offloading mechanism for smart cities," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 30–37, 2017.