ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/

An Efficient and Load Balanced Task Offloading in Vehicular Internet of things

B. Ravaei[1,*], S. Ravaei[2], S. MoshrefZadeh[1], O. Rahmani Seryasat[3]

[1] Department of Computer Engineering, Yasouj University, Yasouj, Iran.
[2] Department of Midwifery, School of Medicine, Yasouj University of Medical Sciences, Yasouj, Iran.
[3] Department of Electrical Engineering, Faculty of Technology and Engineering, Shams Higher Education Institute, Gorgan, Iran.

ARTICLE INFO

Article History:
Received 5 January 2022
Received in revised form
30 February 2022
Accepted 27 March 2022
Available online 29 March 2022

Keywords:
Task Offloading, Internet of Things, Vehicular Internet of Things, Load Balancing

ABSTRACT

In Vehicular Internet of Things (VIoT) environments, vehicles with limited computational resources often need to offload tasks to other vehicles or edge servers with surplus capacity. However, the highly dynamic and mobile nature of VIoT networks poses significant challenges to guaranteeing timely and efficient task offloading. To address this, we propose a novel approach called Vehicular Internet of Things Task Offloading (VIoT-TO), which partitions the network into a cellular structure and employs reinforcement learning to determine optimal task offloading strategies. Specifically, the system learns how to identify nearby idle task servers either peer vehicles or edge servers within each network cell. The proposed method leverages Q-learning to solve the reinforcement learning problem, with the task offloading modeled as a Markov Decision Process (MDP). The reward function is carefully designed to encourage fair load distribution across servers while also prioritizing servers that are geographically closer, thereby reducing communication latency. As a result, the overall task offloading delay becomes more predictable and manageable. Experimental evaluations demonstrate that VIoT-TO outperforms existing benchmark approaches in terms of task offloading delay, load balancing efficiency, and task completion rate. These findings suggest that VIoT-TO is an effective and scalable solution for real-time task management in vehicular networks.

1. INTRODUCTION

The current trend in automotive driving leads to increase the number of sensor in vehicles, which causes an enormous volume of intra-vehicle and inter-vehicle data communication and introduces a new framework named Vehicular Internet of Things (VIoTs) [1]. This trend in VIoTs also causes the necessity of enormous computation resource to process sensor data, network data and apply AI algorithms in order to tackle the delay sensitive tasks [2].

* Corresponding author: ravaei@yu.ac.ir
Department of Computer Engineering, Yasouj University, Yasouj, Iran

http://dx.doi.org/10.47176/TMI.2022.46

Since the vehicle local computation power is limited, one of practical measure is to migrate task from the local vehicle to other devices to overcome computation intensive and delay sensitive tasks.

These delay sensitive and computation intensive tasks have central role in vehicle applications to achieve a safe, efficient, and comfortable experience, which are initiated by the driving systems or vehicles' passengers. As these vehicle applications are computation-intensive and delay sensitive, then they expected a significant computation resources [3, 4], which is usually not available on local vehicle.

If the current trend in autonomous driving continues as present, then the future vehicles will be equipped with various sensors (for in-vehicle and external environment sensing) to gather environmental and internal data and then some sophisticated computation should be done on this data to accomplish autonomous driving, and guarantee the safety requirement. The enormous amount of gathered data cannot be handled by a node locally, and it is necessary for vehicles to perform the computation in collaborative way and also exploit edge and cloud server to help the inadequate internal resources of on road vehicles. The approach of computing vehicular applications remotely by other nodes, edge servers or cloud is called as task offloading. By considering the application requirements and shortage of local resources, it is essential to offload or transfer tasks to other vehicles or tools [5]. The concept of task offloading is a solution to the problem of insufficient local resources by transferring the computation-intensive and delay-sensitive tasks to other vehicles (V2V) [6], infrastructure (V2I) [6], and everything (V2X) [7] having idle resources.

Task offloading in VIoTs not only has challenges of traditional offloading, but also exposure to high mobility of vehicles [7]. Therefore, any proposed method should consider the mobility of vehicles and has a solution to overcome the highly dynamic environment of VIoTs. To fully address the task offloading in VIoTs some challenges should be resolved, such as cooperative vehicular environment, interference and heterogeneity, connectivity and reliability, offloading AI algorithms, task partitioning, migration of computing tasks, evaluation criteria, and finally security and privacy [2].

In this paper we just focus on connectivity and reliability, and propose a solution for finding a routing approach to efficiently offload tasks in VIoTs. The proposed method wants to find a communication solution that satisfies the latency and reliability requirement of VIoT applications. The VIoT applications such as safety, driver assistance system, and platooning have some unique and rigid requirements. These applications communication requirements are a foundation for implementing other aspects of task offloading.

For task offloading in VIoTs, have three different potential opportunities consists of cloud computing [2], edge computing [2, 8-11], peer vehicle computing [2, 8, 9] have been proposed in literatures. The cloud computing is a centralized approach and it experiences unstable connection and long delays and it is not appropriate for delay sensitive task [2]. Another opportunity for seeking computation power is the edge of VIoTs, in which some edge devices can provide resources in proximity of VIoTs to offer low-latency task offloading solution [2, 8]. The final opportunity is the computation power of other on-road vehicles which have high computing resources [2]. In this paper, we just consider other nodes and edge computing and respectively called them as peer vehicle (PV) computing and edge server (ES) computing, and we call both of them as task servers.

In PV computing approach, each vehicle may have some unused resources which can be shared to other vehicles. There are some advantages in sharing unused resources of vehicles together, but by far the most important are the potential to work in disastrous situation and alleviate the high communicational and computational burden on edge resources. In this approach, parked vehicles or vehicles with extra resources can play the role of task server and facilitate task computation of other vehicles [2].

Another approach in task offloading is to utilize edge servers which can be deployed at access points such as 4G or 5G base stations or roadside units (RSUs). In this approach, the edge computing facilities are located in vicinity of mobile user to provide required services and applications with low latency and suitable reliability. Moreover, task offloading on edge servers improves the computing efficiency and reduces the communication bandwidth and consequently enables the task processing delay. Therefore, edge server deployment can be a bridge between the limited resources of vehicles and growing resource requirement of vehicles [2, 11].

Current approach to tackle the task offloading problem have major drawbacks. The most important ones include centralized approach to assign resources and manage task offloading, do not consider application latency-sensitivity

requirement, the limitation in edge server computation capacity is not considered, and some kind of oracle knowledge is demanded.

Therefore, we have proposed a reinforcement learning based method named Vehicular Internet of Thing Task Offloading (VIoT-TO) which forward tasks to task servers that have extra computation resource to perform task before their deadlines. The VIoT-TO design utilizes a mechanism to select an idle and near task server that causes the loads distribute between task servers fairly and a special task server do not snowed under with a lot of tasks. On the other hand, the mechanism used in the proposed method prefers the near task server than far ones. Therefore, the proposed method has a suitable performance in comparison with rival methods in terms of offloading delay, task completion ratio and load balancing index.

To sum up, VIoT-TO exploits a reinforcement learning approach in a cellular structure which distribute the tasks between different task servers. In this paper, the main contributions of VIoT-TO are listed as follows:

a) A decentralized approach is used to assign and offload task to task servers.
b) In the proposed method, a server is selected as a host to perform a task by considering the distance and inactivity of it, so the task time completion is less then rival methods.
c) The tasks are distributed between task servers in a fairly manner, which means the load balancing is considered in task offloading.

The remainder of this paper is organized as follows: In Sect. 2 system model and the proposed method are described. Section 3 focuses on evaluation and simulation of method and compares it with rival methods, and finally, the paper concludes with a recap of its main achievement and future works in Sect. 4.

2. SYSTEM MODEL AND PROPOSED METHOD

The core aim of the proposed method is to decide to forward a task from one a vehicle to other vehicles and transfer it to a task server. When a task is generated, if enough computation resource is not available locally the source vehicle decide to offload it to a task server. The first thing for a source vehicle is to forward it to a neighbor vehicle which is the most likelihood to offload it on an idle task server. Once a new vehicle receives the task, it makes the similar decision for performing task or offloading it. Therefore, the vehicle which receives a task is called custodian vehicle and has duty to perform or forward to it to a task server.

In the real world scenarios, vehicle nodes move over a large area which makes forwarding messages hugely complex as well as incurring significant overhead. For example, cabs in San Francisco trace roam through places which may be farther than 100Km apart [6]. Thus, flat and unstructured network architecture cannot satisfy the quality of service requirements of task offloading because the existence of a gigantic number of vehicles leads to complication and performance degradation. So, in this paper we disclose a cellular approach that initially forwards a message to the destination cell and then successfully delivers it to the destination node.

3. CELLULAR STRUCTURE FOR URBAN AREA

The proposed method use a cellular structure to predict the location of idle task server in vicinity and find a route from source vehicle to task servers and offload the task and perform on task server. The proposed cellular structure approach divides the urban area into cells, as illustrated in Fig. 1. Here, the coverage area of each cell is limited by the cell radius α, which is the distance from a cell center to its border, which can be considered as the wireless communication range of edge servers.

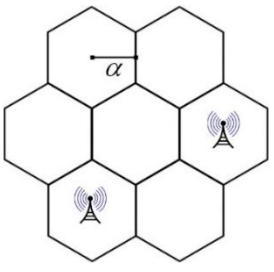

Fig. 1. Cellular structure to divide the network operation area

Since the main purpose of this paper is to design an efficient, load balance and scalable task offloading technique, and in order to boost the chance of successful offloading and completing a task, a cellular model is proposed in which two different type of task servers include vehicle server and edge servers (illustrated in Fig. 1). As can be seen, the cellular structure is not homogeneous. In other words, in some cells a base station (or RSUs) with computing power are installed, which can accept the offloaded task and performs the task. Each cell has a hexagon shape and together they cover the whole network area. As mentioned, all cells are not equipped with infrastructure resource, and they are just regarded as virtual cells to help implementing offloading algorithm. A reinforcement learning approach is used to find a near optimal task server which can perform task and return the result before the task deadline.

The proposed method VIoT-TO is an intelligent method which offloads tasks from task vehicle to task server and return back the result, which considers a certain task as a single message and then routes tasks as a messages between cells instead of routing them between vehicle nodes. Furthermore, if stream of vehicles between cells is also taken into account then we find that there is a regular movement between cells. A cell has limited outgoing links to other cells and stream of vehicles move between two cells. As a result, this stream is more reliable than just one relay vehicle.

To clarify, consider the following scenario: A task is generated by a source vehicle at cell S_1. As show in Fig. 2, three different scenarios may be happened. In the first scenario, a source vehicle has enough resource to perform the task. Second scenario, the source vehicle does not have enough resource to perform task and offload toward a near cell which is likely to be some surplus computing resource to perform task. In this scenario a peer vehicle as a task server with surplus computing power performs task. In third scenario, the source vehicle has shortage of computing resource and forwards the task toward other devices, but there are not any vehicles with surplus computing resources and consequently it finally offloaded on an edge server. Therefore, after forwarding a task for several times, it is delivered to a task server at cell S_n the task server perform the task and return the result of the task to source vehicle. It might the source vehicle transits to new cell S_h and the result should be returned back to source vehicle at S_h. To find a path from cell S_1 to S_n, the inter-cells vehicle stream information is required. To obtain it, the number of vehicles that move between two cells over a given period is used as weight of inter-cells link. It should be added that the amount of vehicle stream between two cells remains almost constant for a long period of time. For example, the number of vehicles that move between an airport and downtown is unchanged for days, weeks or even months, thus the path between different cells can be obtained once and used for a long time. The outcome of route finding decision for source cell S_1 to server cell S_n is similar to $S_1 \Rightarrow S_2 \Rightarrow S_3 \Rightarrow \ ... \Rightarrow S_n$.

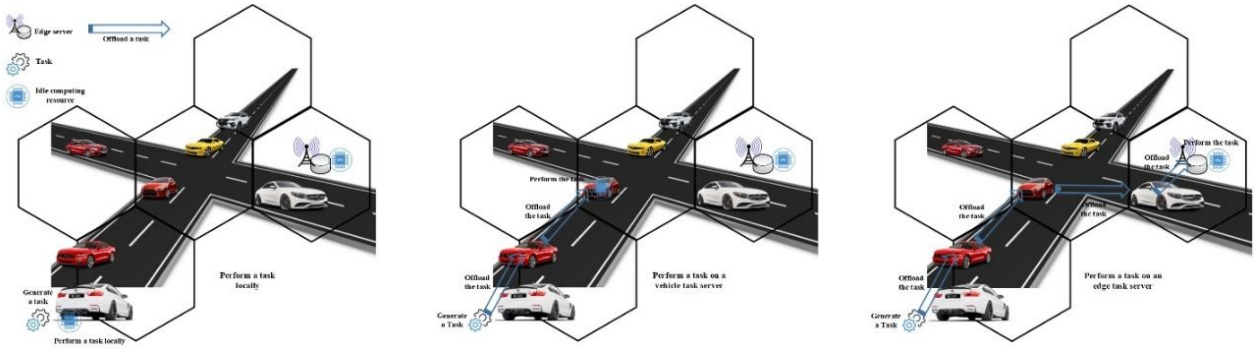

Fig. 2. Three different task offloading scenarios. 1) Perform task locally, 2) Perform a task on another vehicle 3) Perform a task on an edge server

4. TASK SERVER FINDING AND ROUTE SELECTION WITH MARKOV DECISION PROCESS (MDP)

In order to carry a task from a source vehicle to a task server and return the result of a task form the task server to source vehicle, the proposed method which named Vehicular Internet of Thing task Offloading (VIoT-TO) employs a reinforcement learning approach to select server and deliver task to it and return back the result to source

vehicle. Given that the source vehicle must select a vehicle to carry the task, then selecting a proper vehicle becomes a challenge. A simple idea seems to be to select vehicles that go toward the task server. Since it is impossible to determine the next target of vehicles, that idea is not feasible unless an oracle knowledge is available. However, the proposition can be slightly modified into selecting a vehicle that more likely goes toward a near and idle task server. If a relay vehicle has direct communication access to a task server, then the task will be delivered to it, but if the relay vehicle does not have direct access to the task server, then it forwards the task to another relay vehicle to fulfill the task delivery duty. In this case the current relay vehicle is held responsible for the task delivery. After performing a task at a task server, the result should be deliver to source vehicle.

As mentioned before, the main challenging part is to select a relay vehicle to transfer tasks to a task server which is not loaded so much and has some idle resource to can fulfill the task before the task deadline. Actually, the paths to task server are not hard to find, but selecting a relay vehicle to fulfill the delivering task takes some doing. In our method, MDP serves this purpose.

In MDP an agent tries to do a job intelligently and a numerical value from environment is sent to the agent as reward. In MDP, actions affect the current and future received rewards and subsequent environment states. MDP maps the future reward to current reward and tries to maximize the total gained reward which includes the current and future reward [12].

MDP estimates some state-dependent values and uses them to assign credit for short and long-term consequences of actions. The selection of an action is made based on the best policy. As shown in Fig. 3, the agent would learn the best policy by interaction with environment. These interactions include selection of actions by agents and subsequent presentation of new states by environment. Each state is translated to a reward value for agents. Agents try to maximize the long-term reward through choosing the best action in each state.

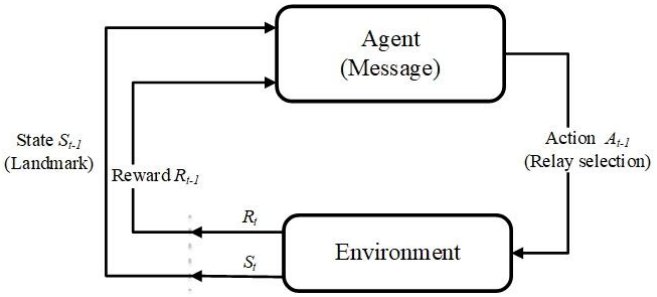

Fig. 3. Agent-environment interaction in a Markov decision process

MDP is a 4-tuple model $\lambda = (\mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R})$ where $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of actions, $\mathbb{P}$ is a transition probability between states, and $\mathbb{R}$ is the received reward after transiting to new states (Table 1).

Table 1. MDP parameters

Parameter	Description and Value
$\mathbb{S}$	States set contains cells
$\mathbb{A}$	Actions set contains outgoing links from a cell
$\mathbb{P}$	Transition probabilities between cells, find by MDP solving algorithms
$\mathbb{R}$	Received reward after transiting to a new state and calculated by Eq. 1

In this paper, cells and selection of vehicles to forward or deliver a task or its result are modeled with states and actions, respectively. Then the probability of dropping a task in a cell is modeled with transition probability. Finally, delivering the task result by source vehicle is modeled with received reward. Here, tasks play the role of agents in MDP. Actually, when a new task is generated, it inherits the policy of selection of the best action. The interaction between environment and a task (an agent) at a sequence of discrete time steps $t = 0, 1, 2, \ldots$ is illustrated in Fig. 3. At each time step t, the state is regarded as current cell $S_{t-1} \in \mathbb{S}$, then based on the state, action (outgoing link) $A_{t-1} \in \mathbb{A}$ is selected. The selected link transfers the task and causes a transition from cell (state) S_{t-1} to new cell S_t and receives reward $R_t \in \mathbb{R}$. The reward function is defined as follows:

$$R(S_t = s) = \begin{cases} 10 & \text{If task is performed by task server in cell } s \\ -10 & \text{If the task is not performed before its deadline} \\ -1 & \text{Otherwise} \end{cases} \quad (1)$$

MDP has the Markov property, which means the state transition and reward probability distribution only depend on the preceding state and action. The following formula is the mathematical representation of probability distribution of state and reward:

$$p(s', r|s, a) = \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}, \quad \forall\, s', s \in \mathbb{S}, r \in \mathbb{R} \text{ and } a \in \mathbb{A} \quad (2)$$

The transition probability which is expressed by (2), is calculated with the historical trajectories of vehicles. As mentioned before, the aim of an agent in MDP is to maximize the long-term reward. In other words, an agent tries to maximize the cumulative reward not just immediate reward. On the other hand, the present reward is worth more than the future reward. Thus, to determine the present worthiness of future rewards, a discounting approach is applied. The cumulative reward can be formulated as follows:

$$G_t = R_{t+1} + \gamma\, R_{t+2} + \gamma^2\, R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k\, R_{t+k+1} \quad (3)$$

where γ $(0 \leq \gamma \leq 1)$ is the discount rate. Thus, a reward that will be received k time steps in the future is worth γ^{k+1} times at present time.

The future reward that an agent can gain completely depends on the subsequent actions it will take. On the other hand, the action that is taken is based on a special strategy that is called policy. The mathematical representation of a policy is a probability of an action selection in a given state. Thus, if an agent is ruled by policy π, then $\pi(a|s)$ is the probability of selecting action $A_t = a$ in state $S_t = s$.

Gaining the maximum reward in MDP requires introduction an optimal policy π_*. This is a policy that performs better than or at least as well as other policies if the expected gained reward for all optimal states is greater than or equal to that from other policies.

Several approaches have been proposed in literature to find or estimate an optimal policy for selecting actions in MDP. In this paper, we use Q-learning algorithm [12] to estimate a solution for the presented MDP. The pseudocode of the algorithm can be found in Algorithm 1. In MDP, two value functions are used to estimate "how good" it is for an agent to select an action or be in a state. The first is state value function which is denoted by $v_\pi(s)$ and is the expected cumulative reward when starting from state s and following the policy π. The formal definition of $v_\pi(s)$\$ is as follows:

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_{\backslash pi}[\sum_{k=0}^{\infty} \gamma^k\, R_{t+k+1}|S_t = s\,] \quad (4)$$

Algorithm 1. Q-Learing algorithm

```
Initialize Q(s, a) ∀s ∈ 𝕊, ∀a ∈ 𝔸
for each episode do
    Initialize S_t
    for each step of episode do
        Choose A_t from S_t using policy derived from ϵ-greedy Q
        Take action A_t, observe R_{t+1}, S_{t+1}
            Q(S_t, A_t) ← Q(S_t, A_t) + β[R_{t+1} + γ max_a Q(S_{t+1}, a) − Q(S_t, A_t)]
        S ← S'
        t ← t + 1
    end for
end for
```

Another value function is state-action value function which is denoted by $q_\pi(s,a)$ which represents the expected cumulative reward of selecting the action a and applying the policy π in state s. The state-action value function is expressed as follows:

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a] \tag{5}$$

The optimal state-value function, v_*, is described as follows and should satisfy the Bellman optimality equation:

$$v_*(s) = \max_a\left(\sum_{s',r} p(s',r|s,a)[r + \gamma\, v_*(s')]\right) \tag{6}$$

In the other hand, $v_*(s) = \max_{a\in\mathbb{A}(s)}(q_*(s,a))$, where $q_*(s,a)$ is the optimal state-action value function which is defined as follows:

$$\begin{aligned} q_*(s,a) &= \max_\pi(\, q_\pi\,(s,a)) \;= \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'}(q_*(s',a'))] \\ &= \sum_{s',r} p(s',r|s,a)[r + \gamma\, v_*(s')] \end{aligned} \tag{7}$$

Since a high computational cost is needed to find the optimal policy, Q-learning as an approximation algorithm is used to find a near optimal policy. To approximate q_* Q-learning define a new action-state value function Q is as follows:

$$Q(s,a) = Q(s,a) + \beta[r + \;\gamma\; \max_{a'} Q(s',a') - Q(s,a)] \tag{8}$$

Where s' is the next state and $\beta \in (0,1]$ is a constant step size parameter. The proposed method trains the MDP for different potential destination cell. So after training phase, it can be applied to find optimal policy for each destination cell.

Therefore, by applying MDP on our model, we would have a policy to select action (outgoing link) in each cell. To sum up, the proposed method initially trains MDP for different potential destination cell scenarios, and then applies the trained MDP to compute an optimal policy for each destination cell, i.e. a policy to select outgoing link for each cell.

By using this model in task forwarding and offloading, a significant improvement in terms of scalability and load balancing can be achieved. One reason for this improvement is that a vehicle does not need complete information about the whole network and idle task servers. Also the applied reinforcement learning approach find out that the idle and near task server are located in which cell and offload tasks there.

5. EVALUATION

In order to evaluate the efficiency of the proposed method and compare its performance with rival methods, the simulator OMNET++ is used to simulate and investigate them. The simulation area is extracted from a 4km * 4km area of San Francisco city then the mobility generator SUMO are utilized to produce traffic in simulation area. to gain insight into the proposed method, we compare it with three benchmark method.

The proposed method is compared with state-of-the-art rival methods adaptive learning task offloading (ALTO) [11], UCB1 [13], and Oracle. The ALTO, is an online task offloading algorithm which is proposed for vehicular environment. The UCB1 method implements a task offloading approach based on classical multi-armed bandit, and the Oracle method is an unrealistic method which all vehicles know complete global information of all task servers, and it is used as a bound. Fig. 4 and Fig. 5 illustrate the average task offloading delay based on task generation ratio and number of task server. The proposed VIoT-TO method outweighs the two rival methods UCB1 and ALTO. For all methods, when the task generation ratio increases the task offloading delay shows a gradual increase. The effect of task generation ratio on Oracle method is insignificant, but for other method is considerable. This phenomenon can be explained by the lack of offloading opportunity by growth of the generation ratio.

Increasing in the number of task servers lead to be more opportunity for task offloading. Consequently, more opportunity of offloading means the custodian vehicles can find candidate servers faster to offload their tasks. As a result, the delay falls down for all methods.

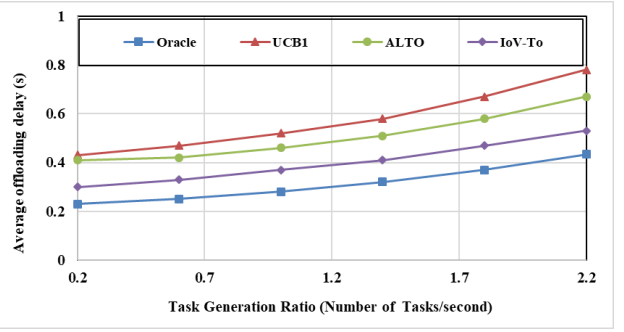

Fig. 4. Task offloading delay based on task generation ratio

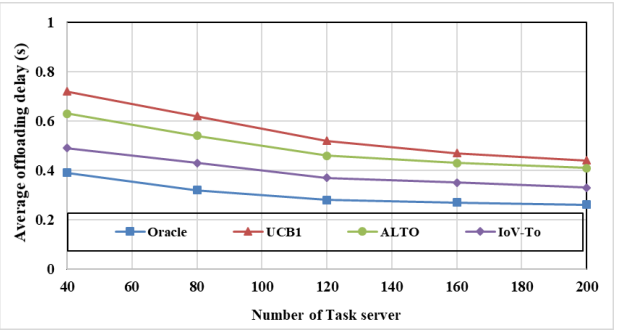

Fig. 5. Task offloading delay based on number of task servers

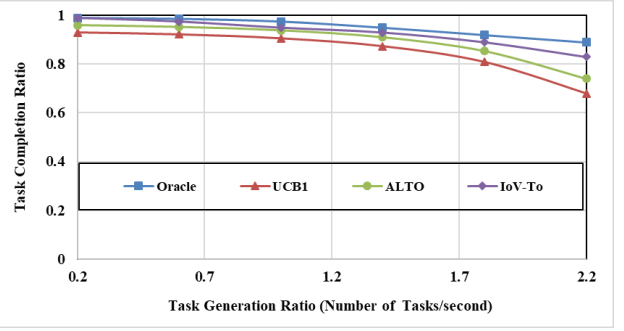

Fig. 6. Task completion ratio based on task generation ratio

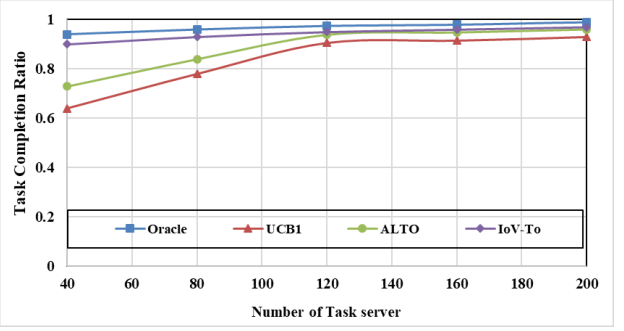

Fig. 7. Task completion ratio based on number of task servers

The effect of task generation and number of task server on delay of our proposed method VIoT-TO shows that it can conform better to variance in offloading opportunity in comparison to its rival methods.

Another important criterion for a task offloading is task completion ratio. Fig. 6 and Fig. 7 shows the task completion ratio less than 0.5 second with respect to the task generation and number of task server. In general, increase in task generation ratio means decrease in the task offloading opportunity and increase in the number of task server means increase in task offloading opportunity. By this interpretation, the more offloading opportunity causes the more tasks will be completed in a predefined time. This pattern can be seen for all methods over various task generation ratio and number of task server. The proposed method has Superior task completion ratio in compression to its arrival methods. A number of reasons have been put forward for this, but by far the most important one is that the custodian vehicle in the proposed method learn how and where to find a near and idle task server, which would have reaching better test completion ratio.

The last but not least criterion for evaluation of proposed method is load balancing index, which is plotted in Fig. 8 and Fig. 9. As shown in these Fig.s, the variation in task generation and number of task servers do not have a significant change in different methods. The load balancing index of our method is close to Oracle method, and far better other rival methods. Since, our proposed method chooses a task server based on its load, and then this leads to a suitable load balance between task servers.

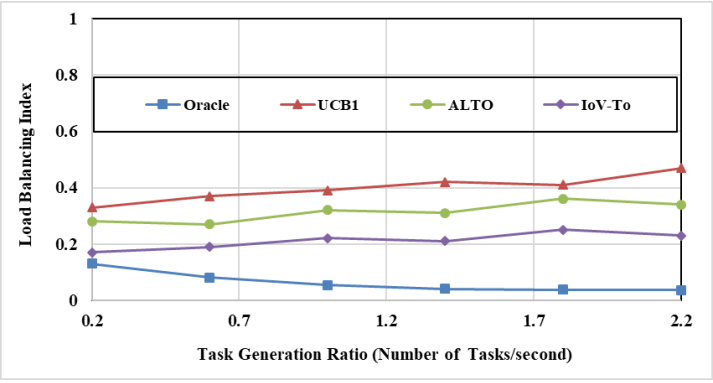

Fig. 8. Load balancing index based on task generation ratio

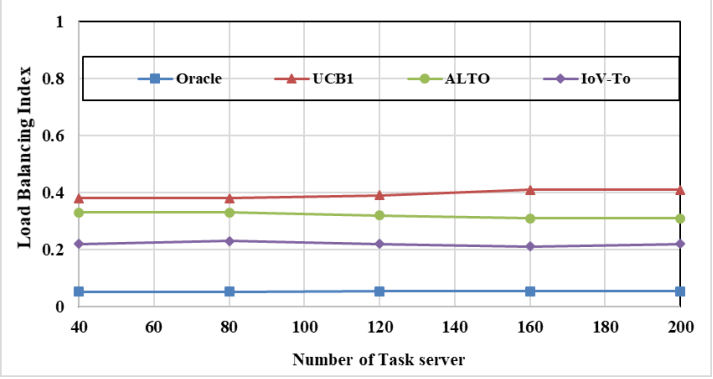

Fig. 9. Load balancing index based on number of task servers

6. CONCLUSION

In this paper, we propose a solution to tackle the communication problem of task offloading in vehicular Internet of Things. The proposed method named VIoT-TO, use a reinforcement learning algorithm and find a near and idle task server. By using these algorithm, tasks are fairly distributed between task servers and consequently the tasks offloading delay is reduced. In addition the result of proposed method in terms of task completion ratio shows that 90% of tasks were completed before half second and which is an appropriate percentage for most of vehicular applications. As mentioned before, we just focus on communication problems in task offloading. While, in the future

work we want to address other challenges of task offloading such as cooperation, AI algorithm offloading, task partitioning and task migration.

Transparency Statement

The data supporting this study are available upon reasonable request to the corresponding author, subject to ethical and confidentiality considerations.

Acknowledgments

We would like to express our gratitude to all individuals who contributed to this project.

Declaration of Interest

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency, commercial, or not-for-profit sectors.

REFERENCES

[1] Du, Z., Wu, C., Yoshinaga, T., Yau, K.-L. A., Ji, Y., & Li, J. (2020). Federated learning for vehicular Internet of Things: Recent advances and open issues. IEEE Computer Graphics and Applications. doi:10.1109/OJCS.2020.2992630

[2] Ahmed, M., Raza, S., Mirza, M. A., Aziz, A., Khan, M. A., Khan, W. U., … Han, Z. (2022a). A survey on vehicular task offloading: Classification, issues, and challenges. Journal of King Saud University - Computer and Information Sciences. doi:10.1016/j.jksuci.2022.05.016

[3] Uhlemann, E. (2016). Connected-vehicles applications are emerging [connected vehicles]. IEEE vehicular technology magazine, 11(1), 25–96. doi:10.1109/mvt.2015.2508322

[4] Singh, P. K., Nandi, S. K., & Nandi, S. (2019). A tutorial survey on vehicular communication state of the art, and future research directions. Vehicular Communications. 18. doi:10.1016/j.vehcom.2019.100164

[5] Boukerche, A., & Soto, V. (2020). Computation Offloading and Retrieval for Vehicular Edge Computing: Algorithms, Models, and Classification. ACM Comput. Surv, 53(4). doi:10.1145/3392064

[6] Ravaei, B., Rahimizadeh, K., & Dehghani, A. (2021). Intelligent and hierarchical message delivery mechanism in vehicular delay tolerant networks. Telecommunication Systems. 78, 65–83. doi:10.1007/s11235-021-00801-1

[7] Ahmed, M., Raza, S., Mirza, M. A., Aziz, A., Khan, M. A., Khan, W. U., … Han, Z. (2022b). A survey on vehicular task offloading: Classification, issues, and challenges. Journal of King Saud University - Computer and Information Sciences. doi:10.1016/j.jksuci.2022.05.016

[8] Sun, Y., Zhou, S., & Niu, Z. (2021). Distributed task replication for vehicular edge computing: Performance analysis and learning-based algorithm. IEEE transactions on wireless communications, 20(2), 1138–1151. doi:10.1109/twc.2020.3030889

[9] Alghamdi, I., Anagnostopoulos, C., & Pezaros, D. P. (2021). Data quality-aware task offloading in Mobile Edge Computing: An Optimal Stopping Theory approach. Future Generation Computer Systems. 117, 462–479. doi:10.1016/j.future.2020.12.017

[10] Dai, P., Hu, K., Wu, X., Xing, H., & Yu, Z. (2021). Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications. Vancouver, BC, Canada. doi:10.1109/infocom42981.2021.9488886

[11] Sun, Y., Guo, X., Song, J., Zhou, S., Jiang, Z., Liu, X., & Niu, Z. (2019). Adaptive learning-based task offloading for vehicular edge computing systems. IEEE transactions on vehicular technology, 68(4), 3061–3074. doi:10.1109/tvt.2019.2895593

[12] Sutton, R. S., & Barto, A. G. (2012). An Reinforcement Learning: Introduction. Mit Press.
[13] Zhou, S., Sun, Y., Jiang, Z., & Niu, Z. (2019). Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks. IEEE communications magazine, 57(5), 49–55. doi:10.1109/mcom.2019.1800230