Towards efficient resource utilization in collaborative fog environment using multi-armed bandit.

By

Rabia Saleem

00000275454

Supervisor

Dr. Asad Waqar Malik

Department of Computing

School of Electrical Engineering and Computer Science (SEECS)

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

January 2022

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Towards efficient resource utilization in collaborative fog environment using multi-armed bandit" written by RABIA SALEEM, (Registration No 00000275454), of SEECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Name of Advisor: Dr. Asad Waqar Malik

Date: 26-Jan-2022

HoD/Associate Dean: ____________________

Date: ____________________

Signature (Dean/Principal): ____________________

Date: ____________________

Approval

It is certified that the contents and form of the thesis entitled "Towards efficient resource utilization in collaborative fog environment using multi-armed bandit" submitted by RABIA SALEEM have been found satisfactory for the requirement of the degree

Advisor : Dr. Asad Waqar Malik

Signature: [illegible]

Date: 26-Jan-2022

Committee Member 1:Dr. Anis ur Rahman

Signature: [illegible]

Date: 26-Jan-2022

Committee Member 2:Dr. Muhammad Shahzad

Signature: M. SHAHZAD

Date: 26-Jan-2022

Committee Member 3:Dr Muazzam A Khan Khattak

Signature: [illegible]

Date: 26-Jan-2022

Dedication

Dedicated to my mother for her never-ending support and my father who has been watching over me from the heaven above.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at National University of Sciences & Technology (NUST) School of Electrical Engineering & Computer Science (SEECS) or any other educational institute, except where due acknowledgment has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my work, except for the assistance from others in the project's design and conception or in style, presentation, and linguistics which has been acknowledged.

Author Name: Rabia Saleem

Signature: Rabia Saleem

Acknowledgments

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You set up in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance.

Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You. I am utterly thankful to my parents for their support, I am nothing but a piece of everything you have wanted me to be.

I would like to thank mu husband and my best friend, Omair Munir for his sheer support and love that helped me achieve my dream.

I would also like to express my utmost gratitude to my supervisor DR. ASAD WAQAR MALIK for his help throughout my thesis and his guidance, tremendous support, and cooperation. I can safely say that I wouldn't have been able to complete my thesis without his in-depth knowledge of this field. Each time I stumbled because of my personal and professional issues; he was there with a helping hand that got me where I am right now in my professional education. There are not enough words to appreciate his patience and guidance throughout my entire thesis. I am heartily thankful to him. His encouragement, guidance, and support from the initial to the final level enabled me to develop an understanding of the subject.

I would also like to pay special thanks to, Dr. Anis ur Rahman, Dr. Muhammad

Shahzad, and Dr. Muazzam A Khan Khattak for being on my thesis guidance and evaluation committee and also for being kind enough to help me whenever I needed their guidance and support.

I offer my regards and blessings to all my friends who supported me in any respect during the completion of the project. Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study

Table of Contents

List of Figures

List of Tables

List of Equations

Abstract

Offloading and resource utilization in vehicular networks and smart cities has been an important problem due to the excessive load on the vehicles despite the availability of multiple resources. These resources are located at the cloud or fog end of the network. Smart vehicles produce a generous number of tasks due to the multiple duties they perform on the road. The tasks that are more CPU-intensive and require real-time results quickly should not wait in the queue to be executed. An efficient offloading technique is required for this purpose that can utilize the resources of the network efficiently while ensuring task execution at a lower waiting time and higher efficiency. There are many existing offloading techniques that have been implemented to solve this problem but none of these techniques have attempted to solve the problem by making the system learn from its behavior. Hence, in our proposed framework, we have introduced an intelligent system of offloading that generated rewards based on certain parameters for each entity included in the offloading decision. Multi-armed bandit is a deep-learning reinforcement algorithm that is implemented on the fog federation. Fog nodes act as both the agent and the arms of the bandit where rewards are assigned to each arm based on different parameters in different variants of the algorithm. The task is offloaded to the highest reward generating fog node after running the algorithm. We have also implemented the network without the multi-armed bandit algorithm and compared the results of 6 variants of the system. The aim of this research is to prove that offloading and resource utilization can be improved if the system acts intelligently by learning from its past behavior and using that knowledge to make efficient decisions.

Chapter 1

Introduction

Background

Cloud Computing

The world of enterprise applications and data is changing dramatically. Data and application integration is now facing evident pressure due to the rapid emergence of cloud computing, the skyrocketing value of the Internet of Things (IoT) data, business users who are getting more empowered by the day, and the fast-paced nature of business. A recent study from Synergy Research Group suggests that operator and vendor sales across six key cloud services and infrastructure market sectors totaled approximately $148 billion in the four quarters by the end of September 2016.
On an annual basis, this indicates that the cloud has risen by 25%. Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) both grew at a rate of 53 percent (PaaS). PaaS accounts for a significantly smaller fraction of that total. Software-as-a-service (SaaS) is the apex of the cloud computing stack, and it's viewed as a replacement for traditional software. The cloud operator provides end-users with an integrated solution that includes hardware, software, and a development platform through SAAS. The key to application software service in science cloud computing is resource management. The optimal decomposition approach is used to address cloud resource allocation so that cloud users' needs are met while cloud providers' profits are maximized [1].
The year 2016 was remarkable for the fact that spending on cloud services surpassed investment in cloud infrastructure hardware and software. Cloud service markets are now growing three times faster than cloud infrastructure hardware and software sectors combined. The forces that brought about the transition have now become systemic. It is feasible to specify new data integration criteria that were previously unknown [1].

Fog Computing

Fog computing is a distributed computing paradigm that has arisen in recent years which brings the cloud's resources and capabilities to incur on the boundary of the network or the fog edges [2]. Because cloud computing is insufficient to handle the vast amount of data generated by the growing number of linked Internet-of-Things (IoT) devices, fog computing has gained popularity. In 2015, there were 15.41 billion devices;

in 2017, there were 20.35 billion devices, and in 2020, there will be 30.73 billion devices [3]. Many challenges arise as a result of such a large number of devices, including network congestion, delay, and privacy concerns if the data is evaluated on the cloud. More specifically, sending data to the cloud uses a lot of network resources, causes congestion, and has a significant delay. Furthermore, sensitive data is created by devices such as home security cameras. Sending them to the cloud over the Internet raises concerns about privacy.

Fog computing is a type of cloud computing that extends cloud computing to end devices in order to better serve time-sensitive, location-dependent, massively scalable, and latency-sensitive applications [4].

Vehicular networks have become an important component of future Intelligent Transportation Systems (ITS) as a result of the development of extensively spread wireless access networks and 5G communication technologies [6]. Vehicles in the ITS are planned to have increased computing power, storage units, communication bandwidth, and sensing power. There are numerous problems in vehicular networking, including vehicle mobility, real-time applications, and connection stability [6], [7]. Because VCC decisions are nearly entirely determined by a remote cloud that is too far away from vehicles, the reaction time of vehicular tasks frequently falls short of the time constraint, especially for real-time tasks. As a result, the Vehicular Fog Computing (VFC) paradigm is offered as an innovation widely for vehicles to serve by making judgments in the local fog cloud [9].

Vehicular Network

As a result of recent breakthroughs in communication technology, new and evolving vehicle communication devices with unlimited features have been created to provide security and wellbeing in the transport industry. VANETs (Vehicular Ad hoc Networks) is made up of a set of communication tools that facilitate moving vehicle units to communicate with one another for a number of reasons [9], [10]. These networks' main purpose is to prevent traffic accidents that result in deaths and injuries. These communication systems are equipped with smart and digital traffic navigation, congestion control, and management technologies to optimize network traffic circumstances. Due to the increased mobility and changing patterns of traffic, communication systems in these networks have been hampered by disconnection, latency, packet loss, and restricted storage limits. Furthermore, radio signals are attenuated, and data transmission is disrupted by environmental impediments. Some of the more expensive services adopted add to the complexity of these networks and communication systems.

Advanced wireless communication technologies such as 3G, 4G, 5G, WiMAX, and LTE are used to provide high-speed internet access along with the highway infrastructure. It provides passengers with unique and creative benefits. Mobile Cloud Computing (MCC) provides a variety of services for VANETs, allowing data to be processed at any time and from any location in the network [11]. In Mobile Cloud Computing, the drivers communicate with the cloud using mobile devices that are connected to the internet. By processing data through utilizing various mobile cloud architectures such as Platform as a Service, Mobile Cloud Computing provides the essential environment for integrating with other technologies for monitoring road safety (PaaS). Mobile devices are limited due to limited battery life, CPU capability, data access, and other computing resources [12]. These products and services are time-consuming and expensive due to real-time information processing, such as traffic jam data processing, message delivery, and accident monitoring.

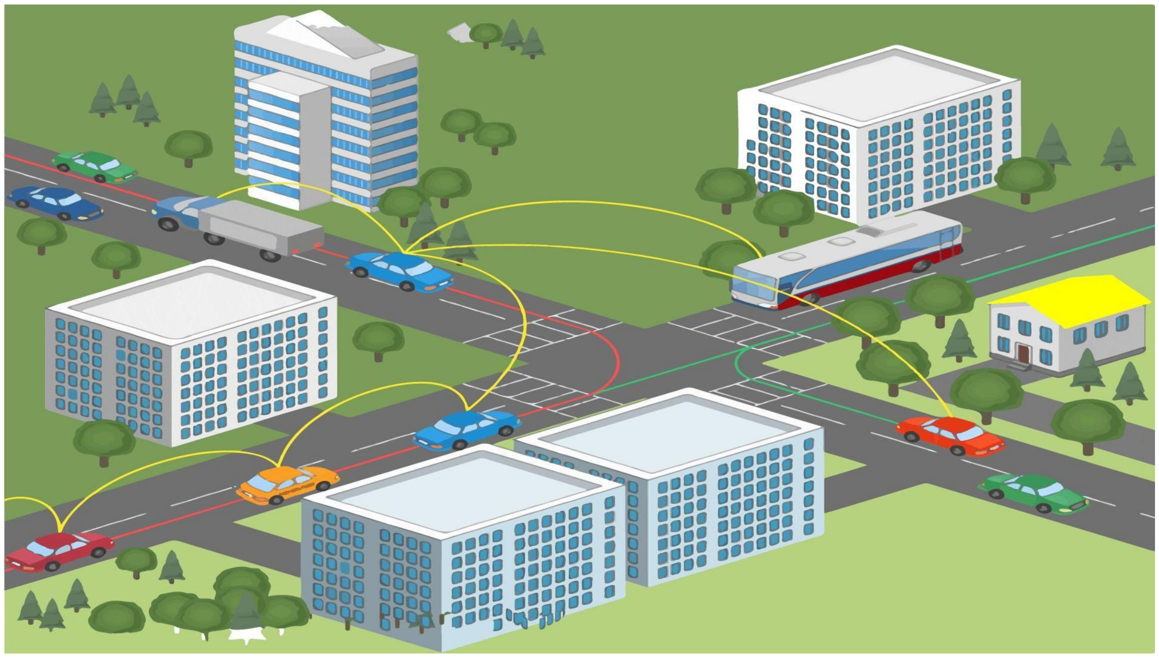


Figure 1. 1: Vehicles connected through a network.

Multi-armed Bandit

In the MAB dilemma, a gambler is given a set of A slot machines (each with one arm) with a predefined payout distribution that is unknown to the gambler. On each play, the gambler selects an arm to play and earns a prize that is independent of previous plays. The goal is to pick arms with the highest estimated cumulative reward over the course of N plays. Finding a balance between exploration (identifying the arm with the greatest expected payoff) and exploitation (finding the arm with the lowest expected benefit) is the key to solving the MAB problem (capitalizing by selecting the arm with the highest observed reward so far) [13].

The multi-armed bandit problem (MAB) belongs to the category of sequential decision-making problems [14]. Clinical studies [15], system testing [16], computer system scheduling [17], and Web optimization [18], [19] are only a few of the uses.

A player is given with K arms in the conventional K-armed bandit format. She chooses to pull an arm kK at each time step t=1,2,... and receives a random payout Rk with an unknown mean k. Over T time steps, the player's purpose is to maximize their expected cumulative reward (or, equivalently, decrease their expected cumulative regret). To accomplish so, the player must find a balance between accurately calculating unknown rewards by pulling all arms (exploration) and always pulling the best arm at the time (exploitation).

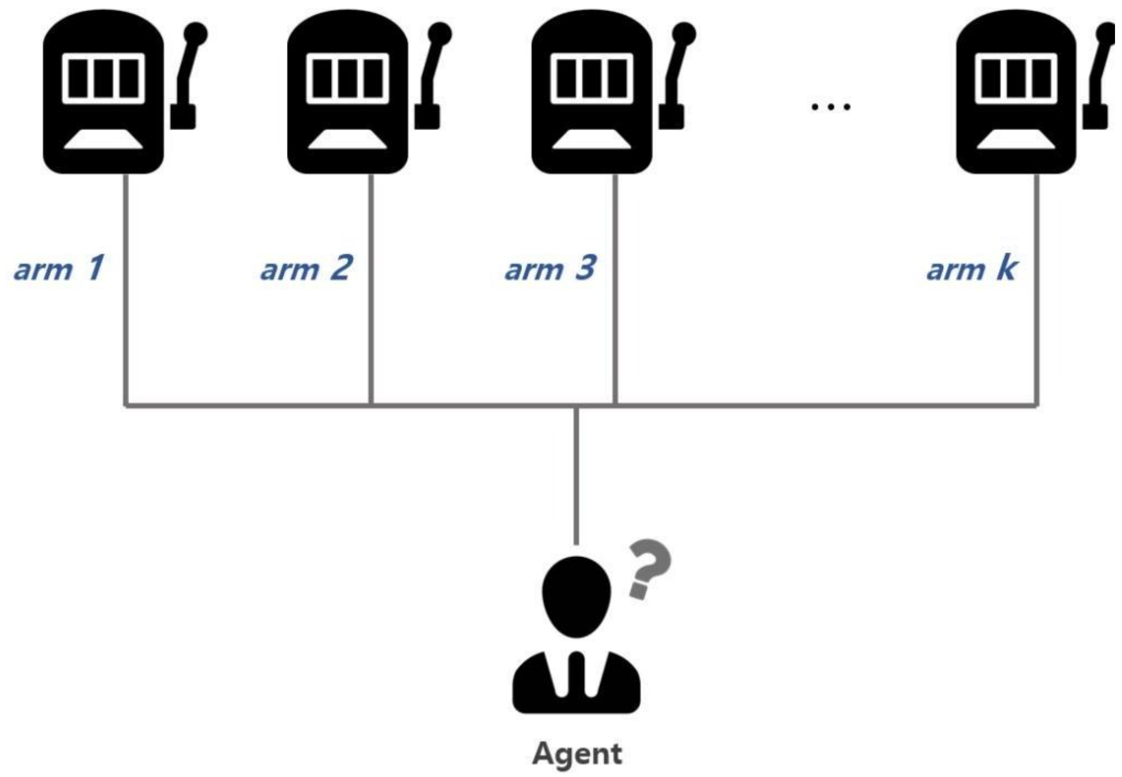


Figure 1. 2: Agent choosing among the available arms.

IoT & Smart City

The smart city is a term used for those cities or areas which aim to make technological developments in the environment to optimize the surrounding functions and to provide a quality life to its citizens [20].

The Internet of Things (IoT) is a system of interconnected objects that form an intelligent network for data transfer that is not dependent on human input [21].

Motivation

To express it, the word data would not be enough to encompass the horizon of big data. Data is exploding at a breakneck pace. Big data is related to all of the fields imagined thus far, not simply computer science, information technology, software engineering, and online web-based systems. Big data is omnipresent, whether it's in medical, sociology, history, education, finance, or other fields. All of the information that a company or organization receives from its customers, as a consequence of a test, or in any other way, must be saved somewhere. The enormous amount of data being generated is rising at such a rapid rate that it is difficult to keep up. Furthermore, data storage is more than just a challenge. There are plenty of additional aspects to consider. Data security, privacy, maintenance, processing, transfer or dissemination, organization, and many more variables are among them.[22]. Google, Twitter, Uber, Facebook, and other companies have a large number of clients, and the data kept in their datacenters is not just about the consumer. However, this data encompasses all of a customer's behaviors while browsing the website, including all of the links he or she clicks, as well as all comments, likes, and reviews, which are all collected and later used to deliver the best possible recommendations to the end-user [22].

The expected increase of data over the next ten years is in the billions of gigabytes [23]. According to Cisco, the monthly data surge is in the tens of Zeta-bytes [24].

Imagine vehicles generating data amongst all these services. Data that is being generated, utilized, and manipulated in a real-time environment needs to be stored and processed efficiently and in a faster way than the data generated by social media. Generating and processing all this information may cause the latency-sensitive tasks their speed and hence the entire vehicle suffers in a smart environment.
Because of technological breakthroughs in smart vehicles, the vehicular cloud is receiving a lot of study interest. Vehicles are expected to become part of a grid network that offers cloud services including processing, storage, networking, and application as a service in the near future. Vehicular cloud and fog computing is a new field that aims to enable applications that are time-sensitive. However, the combination of vehicle networks and fog pr cloud computing poses new hurdles to researchers. To integrate and effectively manage this merging, new frameworks have been developed [26].

Many algorithms have been proposed to cater to the issue of delay sensitivity and efficient decision making in the vehicular networks, however, the scope of parameters in the research are not sufficient to meet the needs of the smart vehicles and the network they constitute. Following problems persist in the prior research.

- The efficiency of task computation comes as a tradeoff with waiting time on the vehicle end.

- Decision-making remains on the vehicle or fog node’s end which increases the delay in the final decision.

Research Objectives

Connectivity and space constraints limit vehicular networks, but more contemporary ad hoc vehicular networks (VANETs) rely on cloud computing. The latter offers low-latency and uninterruptible vehicular cloud computing (VCC) services. However, as services become more common, they require a lot of bandwidth to connect with the cloud server, which is an issue when it comes to meeting QoS requirements.

However, as the number of connected vehicles continues to grow, a new networking paradigm known as vehicular fog computing (VFC) is being deployed to boost computational efficiency [6]. Vehicles use vehicle-to-all (V2X) communication to offload computing to the network's edge.

Furthermore, roadside units can be used for transitory storage (fog nodes). Notably, offloading mechanisms have a communication cost, necessitating innovative design approaches to allocate compute and communication resources in order to balance the energy-performance trade-off while maintaining and delivering a consistent user experience. Even if this cost does not guarantee that the vehicles present in a smart environment like smart cities are making the best decisions. The offloading task has been a problem in the vehicular and cloud networks as the vehicles or any edge device finds it difficult to calculate the best decision in a real-time environment.

We have developed a framework that makes the best possible decision for the fog nodes as they are the ones who receive the task at the time of offloading. If a vehicle is unable to perform a task, it is sent to the fog node for the decision process. The fog node then takes into account various factors to make the best choice. This is where our algorithm intercepts and makes the choice for the fog node.

Following are the objectives of our research:

Efficient Offloading

To reduce the overall offloading cost and time, we model a task offloading system. The task offloading problem in a large-scale network is studied using. This model studies the task offloading problem and constructs a simulated vehicular network to show the performance of the model. If the task cannot be computed at the vehicle end, then it is sent to the fog node for the fog node to decide who gets to perform the task. This offloading mechanism uses a smart deep learning algorithm to offload any required task efficiently.

Low latency

As the algorithm keeps in mind which vehicle has lesser waiting time and where the network congestion may increase in case of increased tasks or workload, it becomes easier for the model to choose vehicles that will perform the task faster. Hence, the tasks are performed at a lesser delay with a lower drop rate and higher efficiency.

Smart decision making

Shifting the paradigm of decision-making from the human being to the machine has been a huge revolution in the recent progression of computer science. Presenting the vehicles with a similar ability helps remove a lot of burden from the vehicle's hardware and the human mind. Multi-armed bandit being deep learning-based algorithms utilize the existing information to predict the best possible results for a problem. Generation of rewards initiates at a random point but as the initial data is gathered the algorithm starts working smartly and ensures robust decision making.

Efficient Resource Utilization

This research has focused on efficient resource utilization in vehicular networks. The algorithm does not burden any single entity with tasks. Similarly, the network traffic avoids congestion by re-routing the task after a certain waiting time has passed for the vehicle. This threshold ensures that one entity is not burdened with tasks and each vehicle gets its fair share of work.

Saving Resources

Many vehicles pass by each other on the road, when one vehicle whose resources are free comes in the vicinity of a vehicle that has all its resources consumed by existing tasks, instead of keeping new tasks in the queue, the free resources of the first vehicle can be utilized to complete a task for the second one. In this way, the resources are being saved for future tasks and being utilized in the best possible way.

Sustainable Development Goals

Following are the Sustainable Development Goals of our research:

Taking the burden off the vehicles

Vehicles carry the responsibility of human life every day. With the emergence of smart cars, the involvement of AI and machines is inevitable in the relationship between vehicles and human beings. Our algorithm ensures that vehicles are not burdened with heavy tasks so that they can ensure the safety of the drivers.

Sustainable cities and communities

Smart vehicles in smart cities allow a better living experience for the people who use them. Having the best decisions made by the vehicles in heavy times ensures the safety and better living standard for the community.

Responsible consumption

Heavy traffic increases pollution and decreases the life quality of both piles of earth and the people living in it. With smart resource utilization and efficient task handling on the vehicle's end, better consumption of energy and resources emerges that was lost with the emergence of vehicles and heavy traffic in the last century.

Utilization of fog infrastructure

The fog ends and vehicular networks exist in smart cities and perform a handful of tasks for the entire community. However, if the resources of these networks are idle, then it is a waste of both infrastructure and energy. This model utilizes the idle resources of vehicles and fog nodes through the connection provided by a fog infrastructure to ensure that essential tasks are being performed with proper resource utilization on the fog end.

Chapter 2

Literature Review

Vehicular networking or VANETs have become a major research area and many algorithms have been proposed to manage and ensure the efficient offloading of tasks in a huge vehicular network.

Our algorithm focuses on offloading the tasks to other vehicles or fog nodes in the vehicle's vicinity. This part will present the related work in this field and how other researchers have proposed to solve this issue of offloading in vehicular networks.

N Lu et al. [26] have explained that ever since the second industrial evolution, vehicles have evolved exponentially, and their importance in modern life cannot be overstated. Vehicles now have wireless communication capabilities for both intra-vehicle and inter-vehicle communications, thanks to significant technological breakthroughs in Intelligent Transportation Systems (ITS) technology. ITS technologies can be used for a variety of applications, including traffic safety, location-based services, and smart transportation.

Vehicles continue to communicate with one another in a network and hence if a task comes that needs to be offloaded to another vehicle, the vehicles can save their resources and energy by offloading that task. However, W Lyu et al. [27] have explained that Offloading, adds significant expense because it necessitates communication between the devices and the cloud. The increased communication has an impact on both energy usage and latency, deciding to offload a challenge. This challenge should be met efficiently to ensure that the resources are being utilized and tasks are being offloaded successfully.

W. Zhang et al. [28] [29], [30] have attempted to solve the problem by allowing users to choose between local and remote execution, with each user deciding whether or not to offload independently of the others. Regardless of how many tasks are being conducted at the same time, it is expected that the cloud will always have enough resources to accommodate the offloaded jobs without delay.

A Bozorgchenani et al. [31] have offered an online and an offline policy method, both of which have two phases: decision making on where to offload and job offloading. The vehicle (or the vehicular user) in both methods makes use of historical offloading records. While the vehicle adapts its decisions over time in the online algorithm, the decision is made once for a set amount of time in the off-policy approach. In the event of unpredictable or non-stationary traffic loads at the edge computing servers, a vehicle can execute an efficient network selection for compute offloading using the provided strategies.

On the Internet of Vehicles (IoV), X Wang et al. [33] have presented a fog-cloud computational offloading algorithm to reduce both vehicles and compute facility power consumption. After establishing the system model, they formulate the offloading problem as an NP-hard optimization problem. Then, to gradually solve the offloading problem, they present a heuristic algorithm. To get the best workload distribution, they have created a predictive combination transmission mode for cars and a deep learning model for computational facilities.

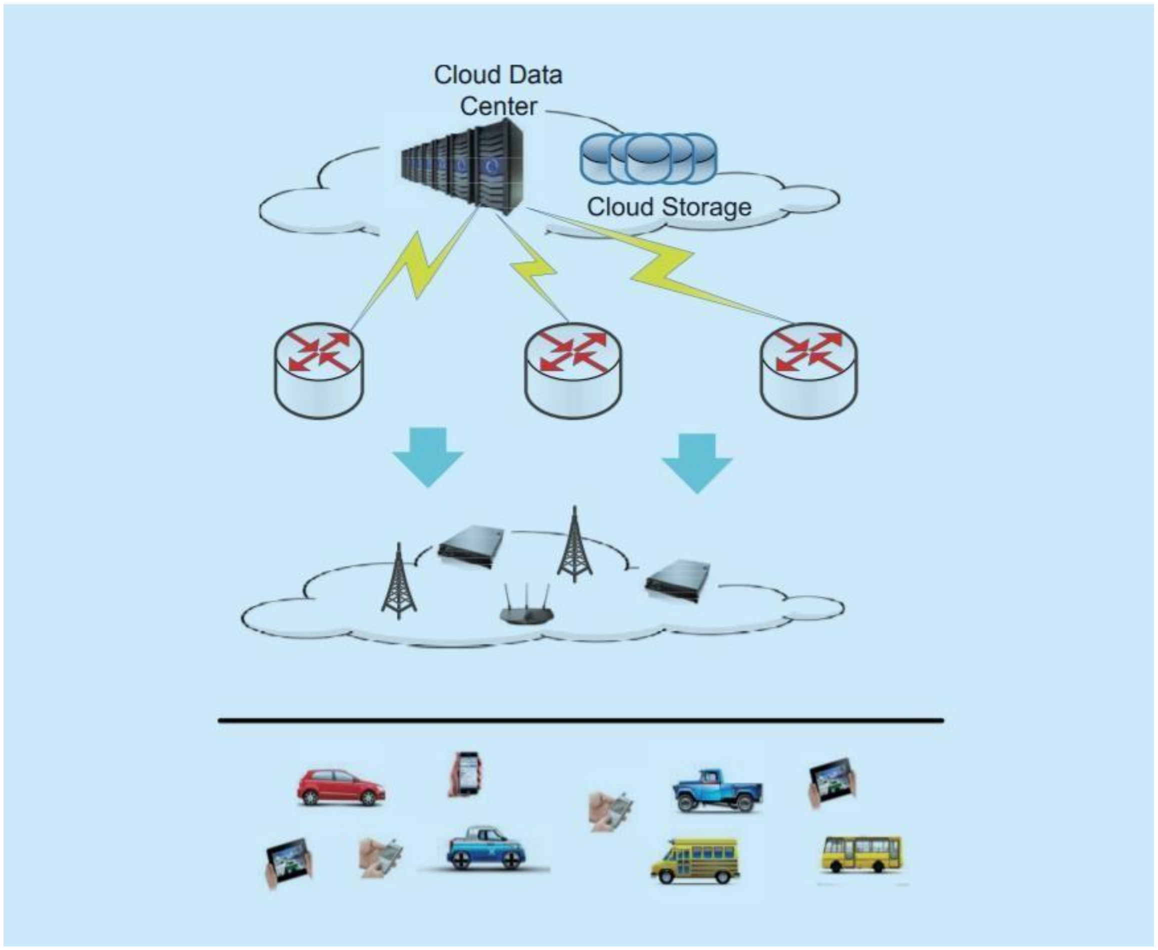


Figure 2. 1: MAB working on the fog nodes.

Ahmed et al. [33] Explain how, as technology advances and smart vehicles and smart cities become more common, each vehicle will be able to link with other vehicles directly or through the ad-hoc networks. Consequently, time-sensitive data can be transmitted through such platforms. However, information reach may be limited in an ad-hoc situation where no relay vehicle is available. Furthermore, crucial information must be delivered within a certain time frame; thus, timely message dissemination is critical. In VANETs, existing data dissemination strategies such as broadcast or partial broadcast generate a high quantity of messages. As a result, broadcast-based systems might produce congestion because all recipients re-broadcast the message, and vehicles receive numerous copies of the same message. Furthermore, due to channel congestion, re-broadcasting can impair the coverage delivery ratio. Furthermore, the usual cluster-based method is ineffective.

Shah et al. [34] present In their research, they describe a data dissemination approach that employs a temporal barrier mechanism to reduce the overhead of messages that can clog the network. The proposed solution is based on the concept of a super-node that transmits messages quickly. Additionally, the time barrier technique has been modified to address this issue in order to minimize unwanted broadcast, which can lead to the problem of broadcast storms. As a result, only the farthest car rebroadcasts the message, allowing it to travel further. Hence, the message can reach the farthest node in less time, improving coverage and lowering latency.

Z. Ning et al. [36] discuss that fog computing, which extends computational capabilities to the network front end, is a viable paradigm for overcoming the aforementioned challenge. It also has the potential to ease cars' enormous computing strain. Where it can be seen that the fog nodes reduce power consumption by offloading the cloud's task. Geo-distributed fog devices, can also quite visibly, lessen message transmission delays. The benefits of fog computing are listed above. However, offloading all jobs to the fog layer is unfeasible since the fog-only model's calculation capability is insufficient to cope with delay growth under heavy workload, and some complicated computational processes must be offloaded to remote cloud servers, as highlighted by W. Hou et al. [37]. As a result, making efficient offloading decisions is crucial in order to reduce the power consumption of computational facilities and vehicles with a delay constraint at the same time [37].

An offloading technique based on a Bayesian classifier was discussed by Fan Jiang et al. [39] Initially, the Bayesian classifier is utilized to classify the task based on its various delay and power usage requirements. Based on the classification results, each vehicle user equipment (VUE) picks the appropriate unloading mode. The task will be offloaded to other cars by using vehicle to vehicle (V2V) offloading mechanism if the VUE's energy consumption needs are higher. Alternatively, it will offload the job using the mobile edge computing (MEC) offloading option. They characterize the offloading and resource allocation scheme as a non-linear issue with the goal of achieving a trade-off between task execution delay and power consumption through offloading option. To arrive at an estimated outcome, a Q-learning-based approach is proposed.

Roadside units (fog nodes) have also been discussed in the study of Saleem et al. [39]. Cars or any other smart vehicles carry several forms of data in vehicular networks, which must be offloaded to Roadside Units (fog nodes) via Vehicle-to-Infrastructure (V2I) message exchange when vehicles enter their areas of coverage. Vehicles have sporadic connectivity with fog nodes because fog nodes are not widely deployed. Vehicles may carry urgent data that must be offloaded to fog nodes as soon as possible. As a result, for data offloading in-vehicle networks, the Quality of Service (QoS) provisioning is critical. With QoS provisioning that employs three QoS functions: overload management, traffic categorization, and admission control, in their study, they offer V2I-Q, a V2I data offloading technique. The information is divided into three groups: low, high, and medium traffic. The fog nodes are kept from getting overwhelmed by overload control, allowing them to receive high-priority data as fast as possible. To accept high-priority data from new vehicles, fog nodes can utilize admission control to stop servicing older cars and discard low-priority data. To accept high-priority data from new vehicles, fog nodes can utilize admission control to stop servicing older cars and discard low-priority data.

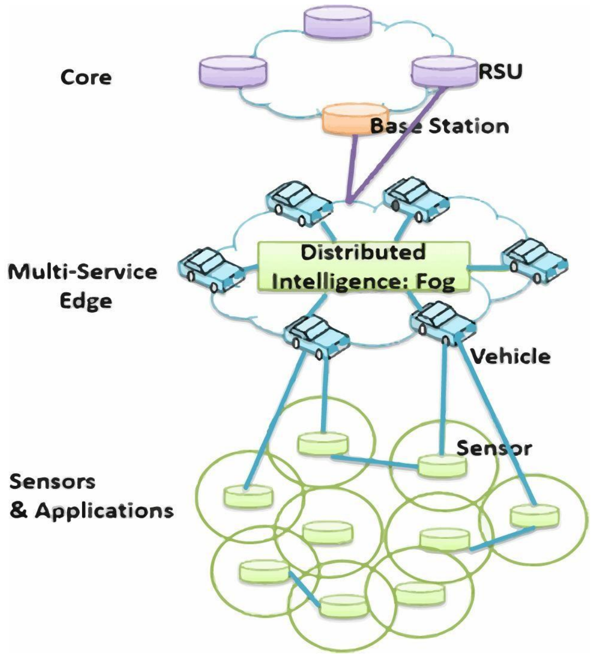


Figure 2. 2 Distributed Fog Network

Lee et al. [40] have highlighted in their work that As a result of the impending 5G network, which will provide low latency for real-time network services in smart factories, autonomous vehicles, and other applications, the number of mission-critical devices is growing. Because edge computing is the nearest to mobile devices and provides the lowest latency and computation energy consumption, a distributed cloud computing system is critical for processing numerous mobile devices. In this study, they investigate autonomous vehicles using video live streaming services. Transmission latency of fewer than 10 milliseconds is required for vehicles particularly. They provide a deep reinforcement learning-based service chaining offloading option for reducing latency while conserving energy. The device's tasks are broken down into service function blocks, each with its own set of responsibilities. As a result, it may simultaneously execute partial offloading and user association in the vehicle's On-Device Edge and the SBS. With the Actor-Critic model, they can get good service chaining offloading decisions for an autonomous driving system.

However, they only take into account the video live streaming service. As a result, there are some limits to calculating various network requirements from the car at the same time, such as sensors, video, speech, and so on. They can't be responsible for the time it took the vehicle to choose task offloading or job transfer to the On-Device Edge or SBS. This argument, on the other hand, considers task waiting time.

Zhou et al. [41] examine a fog network-based vehicle offload approach that isolates computing workloads to achieve maximum resource efficiency while anticipating automotive mobility to reduce offloading latency. By anticipating the vehicle's movement, a model-based reinforcement learning technique was employed at the time to reduce the offload delay time.

A dynamic task offloading option approach has been proposed by Huang et al. [42] for flexibly managing the sub-tasks that originate at the intersection of a vehicle and mobile edge computing. They also developed a system for distributing the transmission queue and computational density of each vehicle by maximizing the deployment of computational power for mobile edge computing.

Finally, Wang et al. [43] argue that sophisticated technologies should be used in vehicular networks to ensure personal safety, prevent traffic accidents, and reduce traffic congestion. Multi-access edge computing (MEC) is a viable solution for addressing such difficulties by using the processing capability at the network edge. Partial offloading, as opposed to standard full offloading, provides more flexibility in terms of the application and deployment of such systems. As a result, in their study, they look at the use of partial compute offloading in in-vehicle networks. They have turned the structure of numerous new applications, such as augmented reality and online games, into a sequential multi-component model by studying their structure. They have also extended the optimization issue from the single-vehicle computing offloading (SVCOP) scenario to the multi-vehicle computing offloading (MVCOP) situation by taking various constraints into account, with the goal of reducing application execution delay. As a solution to this challenge, a deep reinforcement learning (DRL) based algorithm is proposed.

This thesis proposes a deep learning-based solution model for the optimum resource optimization and task offloading of the real-time tasks that are generated by the vehicles.

Chapter 3

Problem Formulation and Proposed System Design

In this study, we have used simulation to generate the results of our proposed solution. The simulation network used is the Sumo simulator and the language used is Python.

The system overview has been presented for understanding in the figure. The figure shows that the entire system is composed of a smart city that has numerous vehicles, 4 fog nodes, or road-side units and thousands of tasks are being generated by the vehicles for the fog nodes to handle and offload. All the entities are connected by a fog-enabled environment. The vehicles can communicate with the fog nodes and vice versa.

The simulation of the network shows that the vehicles generate a minimum of 86 thousand tasks in one simulation. These tasks are either executed locally or being offloaded to the other fog nodes or vehicles. The algorithm focuses on the fog nodes part of the system.

As the vehicles generate the task, many parameters can be taken into consideration while dividing whether to offload it or execute it locally. The vehicle can decide based on the queue list waiting to be executed or the execution time of the tasks. The parameter that we have chosen is the waiting time of the vehicle. If the waiting time exceeds 3 then the task is added to the queue. Then it is offloaded to a nearby vehicle or a nearby fog node based on the availability of the resources.

At the time of task execution, the vehicle is located at a certain distance from the fog node. After the task has been removed from the queue by a fog node, the distance of the vehicle from the current fog node that is holding the task to be executed is calculated. The distance of the vehicle and fog node is calculated by using the following formula

```
distance = sqrt( ((this.x_axis - SrcVeh_object.x_axis)**2) +
((this.y_axis - SrcVeh_object.y_axis)**2) )
```

Equation 1

The task is offloaded and executed from a vehicle to the fog node that is located at a distance of 0 to 150, let's call it fog node 1 and the fog node decides to execute the said task by itself, it is considered a direct delivery. If the distance calculated is 150 to 300, then it is known as a one-hop delivery. Similarly, if the distance is 300 to 450, the delivery is called a two-hop delivery. These deliveries have been recorded for each corresponding fog node and presented in the results section. However, after removing the task from the queue, if the distance calculated exceeds then 450, then it is added to the failed tasks.

The number of direct deliveries, one hop deliveries, two-hop deliveries, and failed deliveries have been recorded for each of the algorithms that have been implemented and tested in this work.

The decision that is being taken at each fog node at the time of receiving or offloading a task, whether to keep it, offload it and where to offload it, is the dilemma in this situation. The algorithm that is proposed in this work precisely works on the decision-making part of the entire process.

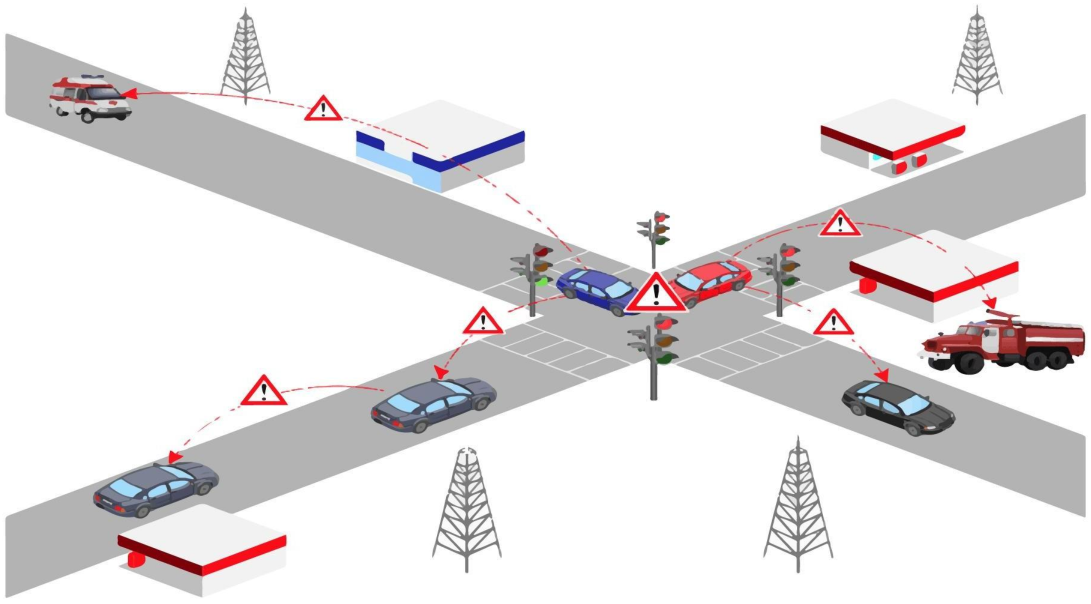


Figure 3. 1: Smart City Communication

Concepts and Entities:

The system entities consist of fog nodes, vehicles, and tasks. The fog nodes compute the fog federation that is offloading the tasks being generated by the vehicles. The system consists of 4 fog nodes snd 422 vehicles in the system.

Fog federation*:*

By regulating traffic flow, a vehicular ad hoc network enhances intra-vehicular networking and lowers traffic accidents. Each vehicle in VANET is believed to have an On-Board Unit (OBU) placed inside the vehicle as well as Road Side Units (fog nodes) installed along the roadside. Sensors, radio communication, and data processing units are all included in the OBU and fog node. V2V communication is when a vehicle communicates with another vehicle nearby, and V2I communication is when a vehicle communicates with the fog node. The Dedicated Short Range Communication (DSRC) protocol is used to accomplish this [44].
The different fog nodes combine to create a fog federation that is going to offload the tasks to aid the vehicle's communication.

MIPS:
The raw processing power of a computer is measured in MIPS. Because measurement techniques vary, MIPS statistics can be misleading, and various computers may require different sets of instructions to execute the same task. "A million Instructions Per Second" is the abbreviation for "Million Instructions Per Second." It's a means of determining a system's processor's raw speed. Because the MIPS measurement ignores other aspects like the computer's I/O speed or CPU architecture, it isn't necessarily a fair way to assess a computer's performance. A computer with a 100 MIPS rating, for example, may be able to do certain functions faster than a computer with a 120 MIPS rating.

CPU:
CPU time (or processing time) is the time a central processing unit (CPU) was utilized to process instructions from a computer program or operating system, as opposed to elapsed time, which includes things like waiting for I/O operations or going into low-power (idle) mode. Clock ticks or seconds are used to quantify CPU time. CPU time is frequently measured as a percentage of the CPU's capacity, which is referred to as CPU utilization. Only when the software uses the CPU to accomplish activities like arithmetic and logic operations is CPU time recorded.

Execution Time:

The waiting time is calculated by simply dividing the MIPS by the CPU value. The formula used in this work is

$$\text{execution} = \text{MIPS} / \text{CPU} \quad \textit{Equation 2}$$

The execution time explains the time the system of a fog node, or vehicle has taken to execute the task. This too can be a parameter to decide where to offload the best task. But the algorithm shows better results with other parameters like waiting time or even random selection. However, to calculate the waiting of the current task, it is important to calculate the execution time first as shown in the formula above.

Vehicles:

Vehicles are the data carriers in our framework. They can receive requests for carrying data either from the data centers or from the data spots. The vehicle upon receiving a data request has a choice whether to carry the data or to drop the request. If it aims to carry the data, then it shall move towards the entity which requested it. The vehicle shall then load the data from the data center/spot to its own storage device. This is done by means of a USB 3.2 wire. Its data transfer rate is 20 Gbps [34], which helps in quick data transfer. Once the data is loaded onto the vehicle, the vehicle is then assigned a data spot where the driver can offload the data. The data spot is assigned to the vehicle based on the destination located nearby. The vehicle again has a choice to offload the data or leave the task. Upon leaving the task, the data will be retransmitted. If the vehicle decides to offload then it moves towards the assigned data spot and offloads the chunk again by means of a USB 3.2 wire. The process is the same in case the data spot requests the vehicle to pick up data from it. In a deployed system in the future, the vehicles can be paid to encourage a larger number of participants.

Execution Of Framework:

The tasks that need to be executed are generated by the vehicle. At each iteration of the code, the algorithm generates a random number that is set to 0.45. If the number is less than this value, the vehicles generate tasks that are added to the task queue.

The tasks that cannot be executed by the vehicles due to the higher load of work are offloaded to the fog federation. The algorithm calculates the distance of the vehicle with the nearest fog node, if this distance is somewhere between 0 to 150, this task is offloaded to that node. This value is variable for each vehicle as for every vehicle the corresponding fog node may be a different one. This fog node's id is stored in "this id". On this node, the framework proposed in this work runs and decides the best choice to offload the task to. This can be the node itself, which means that the node will perform the task by itself, or it can be another fog node. This decision makes up for the best offloading possible in a fog environment. Multi-armed bandit runs a said number of 1000 iterations considering the 4 fog nodes as the arms of the bandit. One of the arms is chosen based on one of the three techniques. Random choice, nearest neighbor choice, or choice based on a list of values that are provided to the bandit.

All three of these methods have been implemented in the work along with 3 methods that do not include the multiple armed bandit for a better approach and understanding of the algorithm. The figure below explains the fog nodes working as agents of the MAB scheme. Each agent chooses the best rewarding arm including itself and offloads the task to that arm.

The Mab algorithm does the following in the framework

- Find the best choice of arm/fog node that the task should be offloaded too.
- Minimize the waiting time of the fog node, the vehicles, and the average waiting time.
- Maximize the efficiency of the fog nodes, the vehicles, and the average overall efficiency.
- Utilize the maximum resources by pushing more tasks toward the fog federation to lower the load on the vehicular end.

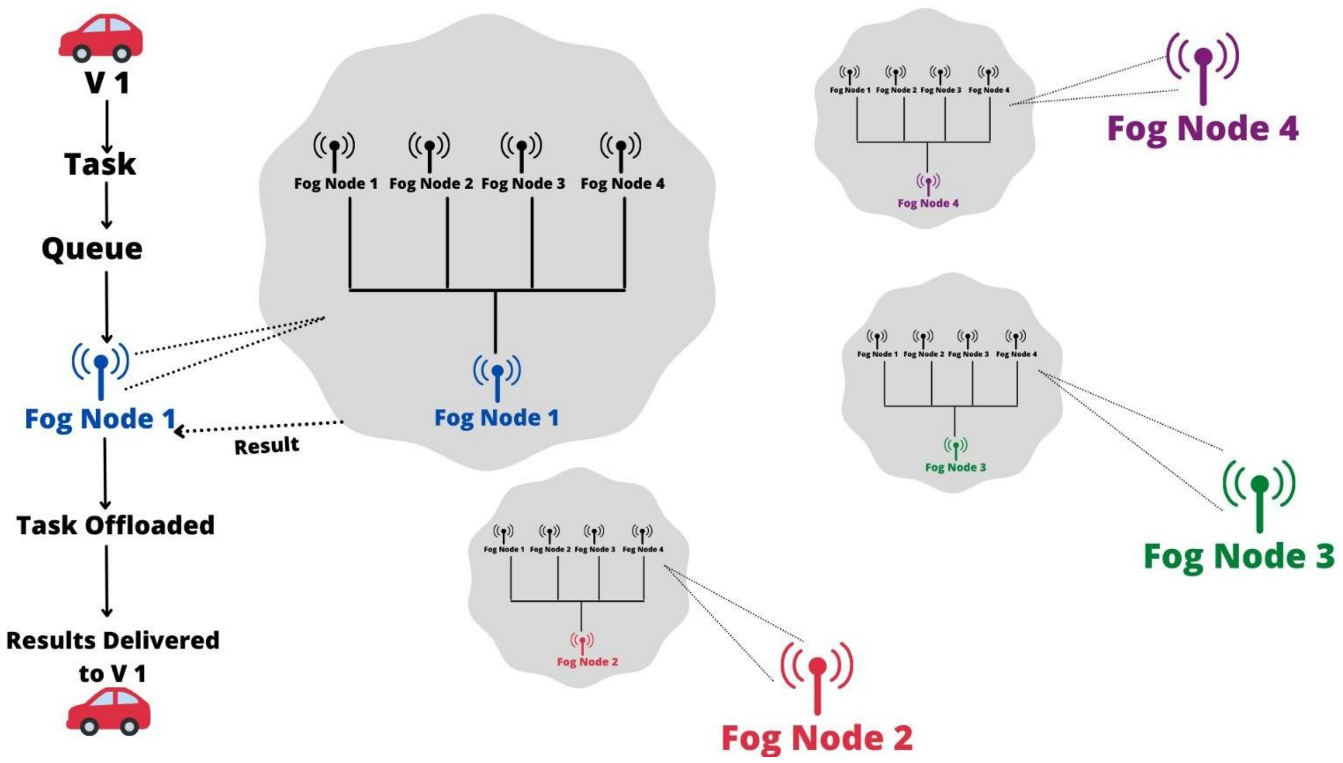


Figure 3. 2: MAB working on the fog nodes.

Problem Formulation:

One of the important challenges to be noted in a smart city is task offloading. As smart vehicles become more common, task offloading is a promising strategy for improving system performance. In other words, the vehicles can pool their computation capabilities to help other vehicles or users complete tasks at nearby compute nodes. It's worth noting that, as fog networks become more complicated, finding an effective task offloading allocation policy becomes more challenging.

Table 3. 1: Nomenclature

Symbols	Meaning
	Road-side Unit / Fog federation
V	Vehicles
WT	Waiting Time
	Execution Time
	Task
dlv_dist	Distance between fog node and vehicle
	Number of arms of the bandit
	Number of episodes of the algorithm
	Number of iterations
	Million Instructions Per Second
	Central Processing Unit
step	Number of times the algorithm runs

There has been extensive research conducted in this area. Many algorithms have been proposed to ensure efficient resource allocation and resource utilization in a vehicular network. A fog federation has extended resources that can be utilized in the network to assure efficient resource utilization and offloading.

Waiting Time:
The waiting time of any task is simply the amount of time the task has to wait in order to be executed. It is simply calculated by the following formulas

$$\text{Waiting Time} = \text{MIPS} / \text{CPU}$$
$$\text{waiting time} = \text{waiting time} - \text{execution}$$ Equation 3

Waiting time plays an important role while configuring a network. The tasks are assigned to multiple vehicles based on the waiting time. If the waiting time of a fog node is higher than the waiting time of the current task at hand, then the task will not be offloaded to that particular fog node. Similarly, before offloading a task, the waiting time of the corresponding tasks and entities is always checked.

Efficiency:

The efficiency of the algorithm is the measure that we check to see how well our algorithm is performing other than the waiting time. Lower waiting time and higher efficiency is the goal of any algorithm that is being applied to the problem.

The formula to calculate the efficiency on the vehicle's side is

E1 = Total computed tasks / Total generated *Equation 4*

The formula to calculate the efficiency on the fog node's side is

E2 = Total_compute_fog node / Total_recv_fog node

The formula to calculate the total average efficiency is *Equation 5*

E = E1 + E2 / 2

The vehicle's efficiency is calculated by dividing the number of tasks *Equation 6*

computed by the vehicle, by the total number of tasks that are being generated by the vehicles.

The fog federation's efficiency is calculated by dividing the number of tasks computed by the federation, by the total number of tasks that are being generated by the vehicles.

The total efficiency is calculated by adding the tasks computed by the vehicles and the fog federation and dividing the number by the total tasks generated.

Multi-Armed Bandit

The multi-armed bandit problem is a classical reinforcement learning technique with an epsilon-greedy agent and a reward-average sampling learning framework to calculate the action-value X(a) to help the agent improve future action decisions for long-term reward maximization.

What we want to do is create a Xt (a) estimate:

Equation 7

$$Xt(a) = E\,[\,Q_m \mid K_m = a\,]$$

When action K_m is taken at step n, Xt(a) is the estimated, expected reward (Q_m). We'll design a model that will iteratively converge on the true value of each action.

This greatest expectation or greedy action is denoted by the letter K*m. This is the exploit side of our previously discussed explore-exploit conundrum, and it makes perfect sense if our goal is to maximize our reward.

$$K_m = \max_m(X_m(a))$$ Equation 8

The ϵ-greedy techniques have a clear flaw in that no matter how many examples they observe, they continue to include random noise. It would be preferable if they could find an optimal solution and continue to use it. To do this, we can use ϵ -decay, which lessens the likelihood of exploration with each step. This is accomplished by formulating as a function of the number of steps, m.

$$\epsilon(m) = 1/1+m\beta$$ Equation 9

Where $\beta<1$ is used as a scaling factor to slow down the scaling pace and give the algorithm enough time to explore. In this situation, we also include +1 in the denominator to avoid infinitesimals.
Lastly, while new information allows the values to converge to their true means, the algorithm explores early on in order to maximize its returns. This method does necessitate the inclusion of some more background knowledge in the setup because we need to have some understanding of the benefits in order to overestimate them.
Hence, for this purpose, we initialize the initial rewards to kick start the algorithm which has shown significant improvement in the overall results.

Potential Objectives:

Followings are the potential objectives of our framework:

1) {}

2) {}

Our aim is to minimize the waiting time in the fog federation and maximize the efficiency of the framework to yield the best results in the performance of the network.

Chapter 4

Simulation Tool

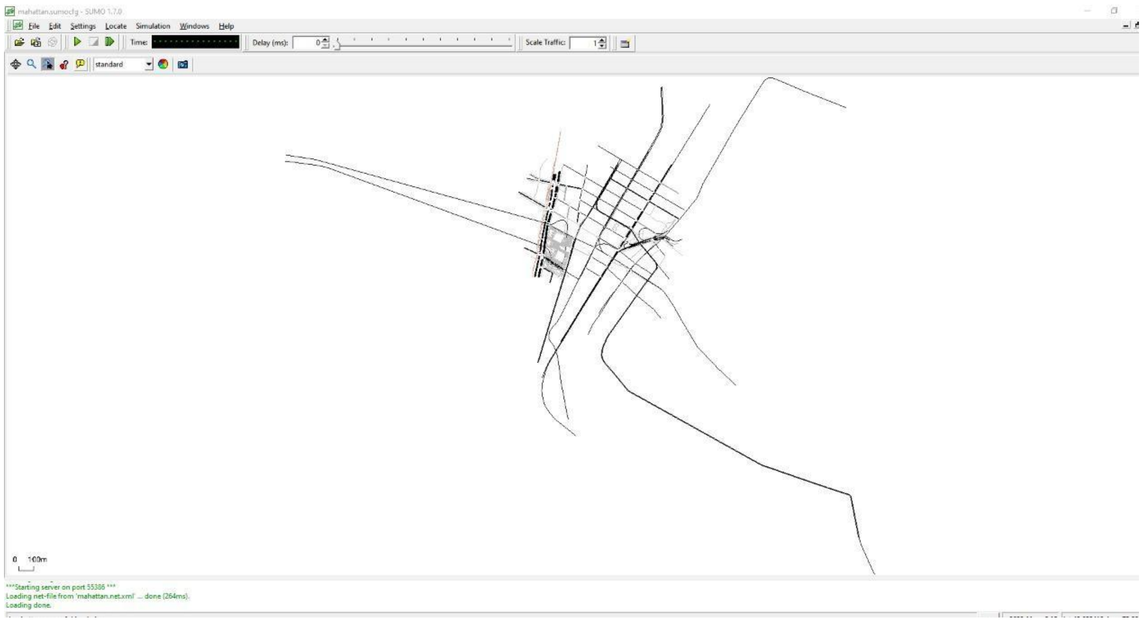
Sumo Simulator:

SUMO is a free and open-source traffic simulation application. This has been available since 2001 and enables you to simulate multimodal traffic networks involving vehicles, public transportation, and pedestrians. Network import, routing calculations, visualization, and emission computation are just a few of the supporting tools available with SUMO, which simplify basic procedures for the creation, execution, and evaluation of traffic simulations. SUMO can be customized via custom models, and the simulation can be managed remotely via a range of APIs.

The framework consists of 4 fog nodes, 422 vehicles and task generation from every vehicle sums up the tasks to somewhere near 87000 which is variable depending upon the waiting time and the execution time of the system. The number of fog nodes and vehicles can be varied based on the algorithm and its requirements.

The size of the data set is around 116000.

The zoomed-out view of the entire network is shown in the figure below.

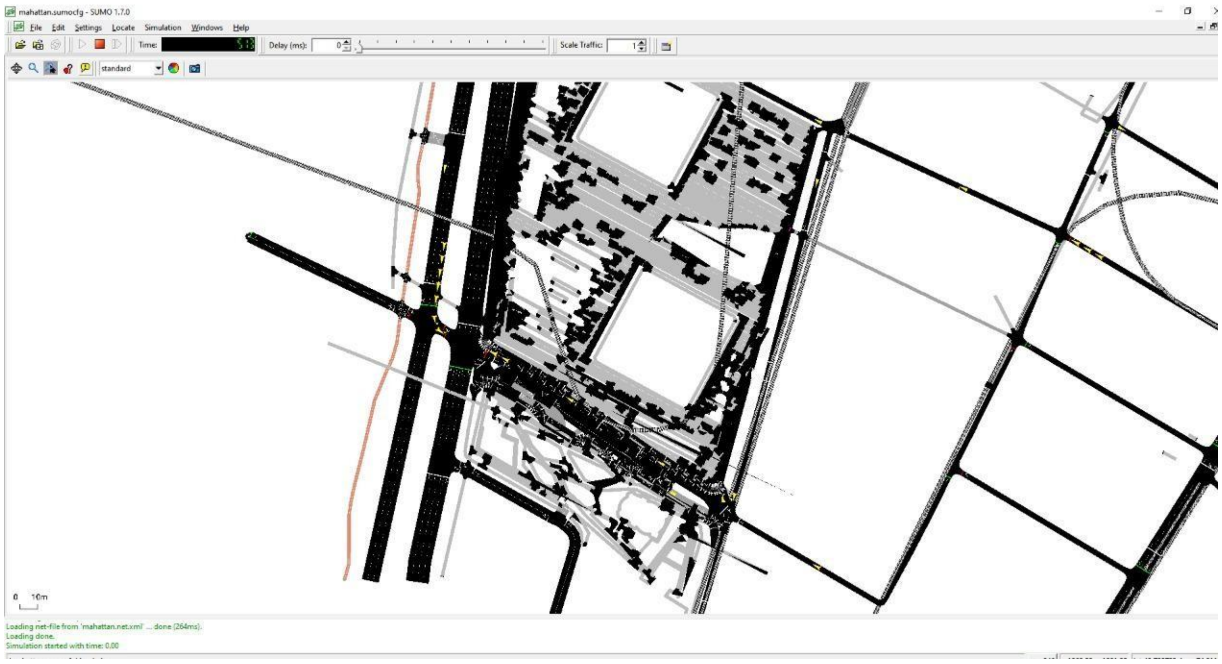


```
***Starting server on port 55386 ***
Loading net-file from 'mahattan.net.xml' ... done (264ms).
Loading done.
```

Figure 4. 1: Sumo simulator implementing the Manhattan data set.

Fig: Simulation showing the Manhattan data set simulated for 422 vehicles and 4 fog nodes.

The below figure shows the zoomed-in view of the simulated network. The yellow arrows represent the vehicles passing by the roads in the network.



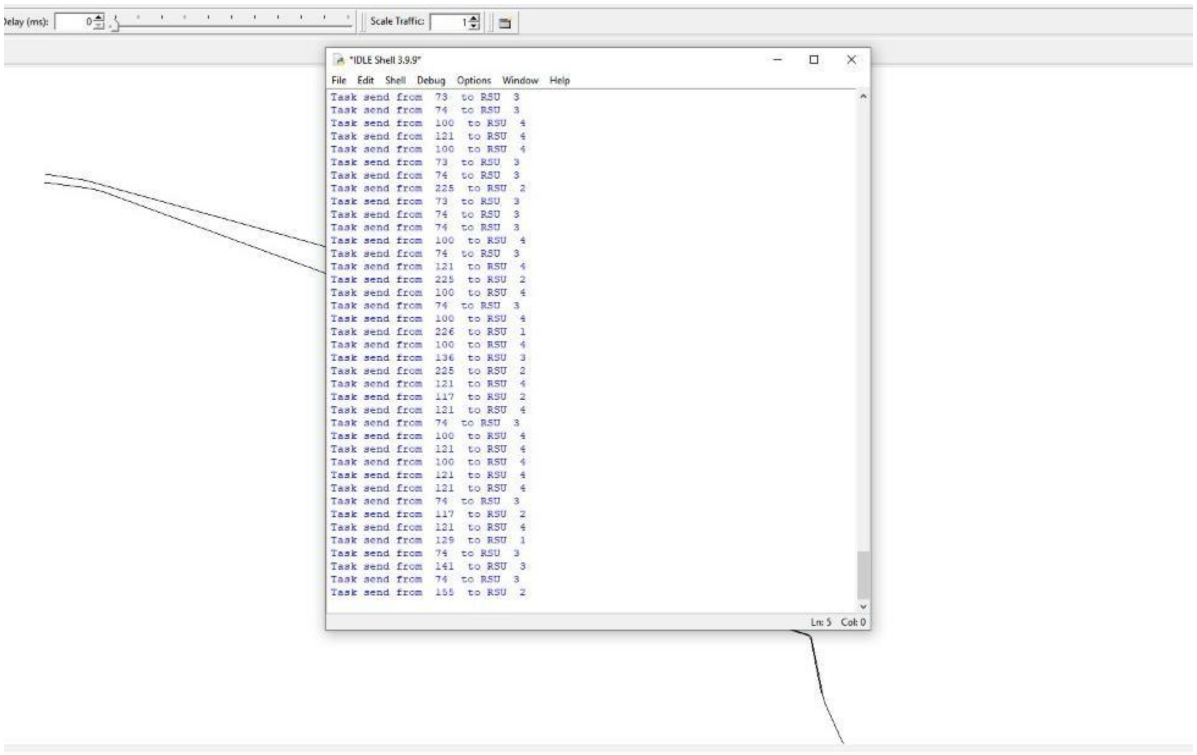
```
Loading net-file from 'mahattan.net.xml' ... done (264ms).
Loading done.
Simulation started with time: 0.00
```

Figure 4. 2: Closed view of the SUMO simulator.

The locations of the said fog nodes have been specified in the algorithm by specifying the x-axis, y-axis and the ids have been assigned to each node as 1,2,3, and 4. Their CPU and MIPS have also been set as 100 each, these values can be varied before running the simulation to check the variations in the data and results.

As the simulation runs the results show how every task is being offloaded. The task id is mentioned with the vehicle that is sending the task to a certain fog node with its id.

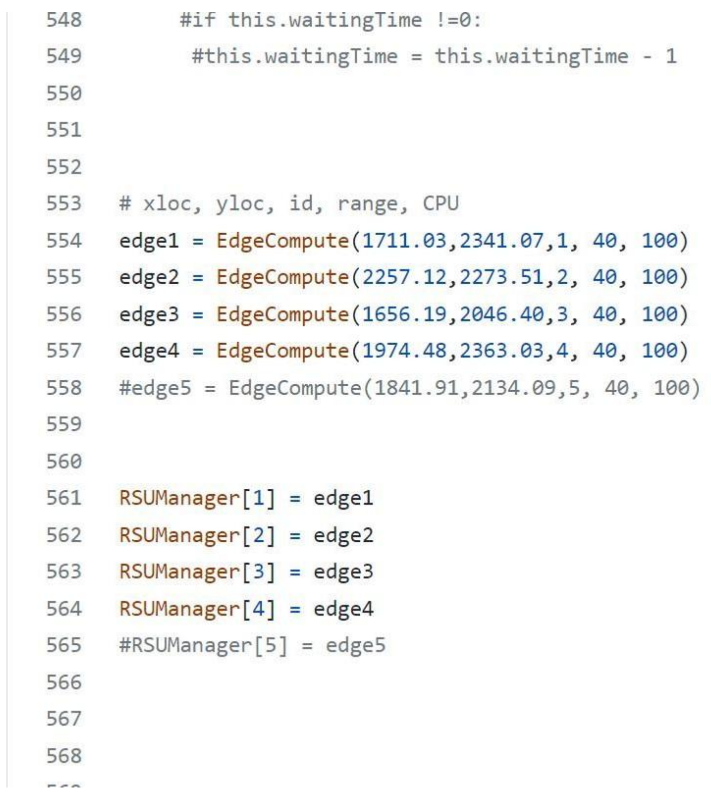
Following is the format in which the simulation shows the tasks being offloaded. This format can be altered by changing the algorithm.



```
Task send from  73  to RSU  3
Task send from  74  to RSU  3
Task send from  100  to RSU  4
Task send from  121  to RSU  4
Task send from  100  to RSU  4
Task send from  73  to RSU  3
Task send from  74  to RSU  3
Task send from  225  to RSU  2
Task send from  73  to RSU  3
Task send from  74  to RSU  3
Task send from  74  to RSU  3
Task send from  100  to RSU  4
Task send from  74  to RSU  3
Task send from  121  to RSU  4
Task send from  225  to RSU  2
Task send from  100  to RSU  4
Task send from  74  to RSU  3
Task send from  100  to RSU  4
Task send from  226  to RSU  1
Task send from  100  to RSU  4
Task send from  136  to RSU  3
Task send from  225  to RSU  2
Task send from  121  to RSU  4
Task send from  117  to RSU  2
Task send from  121  to RSU  4
Task send from  74  to RSU  3
Task send from  100  to RSU  4
Task send from  121  to RSU  4
Task send from  100  to RSU  4
Task send from  121  to RSU  4
Task send from  121  to RSU  4
Task send from  74  to RSU  3
Task send from  117  to RSU  2
Task send from  121  to RSU  4
Task send from  129  to RSU  1
Task send from  74  to RSU  3
Task send from  141  to RSU  3
Task send from  74  to RSU  3
Task send from  155  to RSU  2
```

Figure 4. 3: Tasks being offloaded from vehicle to fog nodes

The positions of all the fog nodes along with their information are shown in the below figure.



```
      #if this.waitingTime !=0:
       #this.waitingTime = this.waitingTime - 1



# xloc, yloc, id, range, CPU
edge1 = EdgeCompute(1711.03,2341.07,1, 40, 100)
edge2 = EdgeCompute(2257.12,2273.51,2, 40, 100)
edge3 = EdgeCompute(1656.19,2046.40,3, 40, 100)
edge4 = EdgeCompute(1974.48,2363.03,4, 40, 100)
#edge5 = EdgeCompute(1841.91,2134.09,5, 40, 100)


RSUManager[1] = edge1
RSUManager[2] = edge2
RSUManager[3] = edge3
RSUManager[4] = edge4
#RSUManager[5] = edge5
```

Figure 4. 4: Class edge compute created the fog nodes which are called fog nodes

Chapter 5

Evaluation and Results

The following parameters and their derivates have been considered while calculating the results and the evaluation of the algorithm. The values have not been mentioned in the table as they are variable for each of the algorithms. These values have been discussed separately below. The parameters, however, remain constant throughout the evaluation. The vehicles generate the tasks which are either locally executed or are offloaded to the fog federation which may intern offload the tasks to the other nodes in the federation. The following parameters have been kept in mind while calculating the results:

Table 5. 1: List of Simulation parameters

Parameters	**Values**
Efficiency at fog nodes	Variable
Waiting Time at fog nodes	Variable
Tasks offloaded to fog nodes	Variable
Failure Deliveries	Variable
Pending Tasks	Variable
Computed Tasks at fog nodes	Variable
Direct Deliveries at fog node	Variable
One Hope Deliveries at fog node	Variable
Two Hop Deliveries at fog node	Variable
Total Task Gen by Vehicles	340 KM
Efficiency at Vehicles	Approximately 86000
Waiting time at vehicles	Variable
The average efficiency of fog nodes and vehicles	Variable
Average waiting time of fog nodes and vehicles	Variable

Algorithms implemented:

Random Offloading

In random offloading, the algorithm is initiated by giving the first fog node id as the random number generated by the algorithm itself. Then every time a task is added to the queue for offloading, the fog node is chosen randomly from the 4 ids, 1,2,3,4. A random library of python language was used to generate a random number at every offloading instance. This algorithm is a purely hit and trial and random approach. It gets no aid from additional knowledge like waiting time or the resources available around the network. It simply generates a random number from the range of 1 to 4 and offloads the task to that fog node.

Sequenced Offloading

In the sequenced offloading, the algorithm chooses the fog node with the id 1, regardless of the position of the vehicle or the fog node. In the next task generation, fog node 2 is selected, similarly, for the third one, fog node 3 is selected. Hence, for every task generation, a sequence of 1,2,3,4 are given to the algorithm that chooses these nodes in the same order. The results generated by this variant have been discussed with other variants below for parameters, waiting for time, and efficiency in fog nodes, vehicles, and overall statistics.

Nearest Neighbour fog nodes

In the nearest neighbor technique, the fog node chooses the node that is placed nearest to it. For every task generation, the node chooses the neighbor that is placed next to it and offloads the task to that neighbor. The results generated by this variant have been discussed with other variants below for parameters, waiting for time, and efficiency in fog nodes, vehicles, and overall statistics.

MAB Offloading with Random Selection

In the MAB offloading with random waiting time, the algorithm runs the bandit algorithm and chooses the best arm that rewards the algorithm, in this case, that arm is the fog node that is passed to the bandit. Among the fog nodes, reward allocation is done randomly for the first time and gradually the algorithm learns in 1000x1000 iterations to select an arm that has the best reward. The rewards are allocated based on the variable “mu”. This variable is set to random by importing python’s random library. The exploitation is made by the algorithm while keeping a decay of 0.1. The best results of MAB were recorded with a decay of 0.1 as compared to 0 and 0.01.

MAB Offloading with Selection Based on Waiting Time

In the waiting time-based offloading variant, the MAB algorithm works similarly as it does in the MAB random, but here the rewards are allocated to the arms based on a list of values provided by us. This list of values is the waiting time. The algorithm works by rewarding the arm that has a lower waiting time as compared to the others. These rewards are allocated for 1000x1000 iterations and the best arm is chosen to pass to the fog nodes for decision.

MAB Offloading with Random Waiting Time

In the mab variant random waiting time, we have included the fog node generated by the algorithm in the above variant "selection based on waiting time" and added it to a list of fog node ids i.e, [1,2,3,4].

If the selection based on waiting time selects an arm from the given arms, we call it 'b'.

So the new list looks something like this,

[1,2,3,4b]

From this list, the MAB algorithm allocates the rewards randomly. What this means is now there is a higher probability for the arm with the least waiting time and highest reward to be selected but it is not necessarily the selected arm, there is still a higher room of data exploitation while choosing the best arm or fog node.

The results for parameters waiting time and efficiency have been presented in the table below and the graphs that explain the behavior of the network have been explained further in this work.

Table 5. 2: Statistics at Fog Node

Statistics at Fog Node	Avg Waiting Time at fog node	Efficiency at fog node	Total Offloaded to the fog node	Pending Tasks at fog node	Computed at fog node	Direct Deliveries at fog node	One Hope Deliveries at fog node	Two Hop Deliveries at fog node	Failure Deliveries at fog node
Random Offloading	**29.62**	**0.93**	**1999**	**129**	**1870**	**351**	**611**	**348**	**556**
Sequenced Offloading	**35.4**	**0.94**	**515**	**28**	**487**	**16**	**276**	**54**	**140**
Nearest Neighbour fog nodes	**54.745**	**0.85639**	**2089**	**300**	**1789**	**340**	**602**	**349**	**494**
MAB Offloading with Random Waiting Time	**61.8**	**0.82**	**2272**	**402**	**1870**	**389**	**569**	**464**	**444**
MAB Offloading with Selection Based on Waiting Time	**27.05**	**0.95**	**596**	**28**	**568**	**31**	**293**	**75**	**165**
MAB Offloading with Random Selection	**13.7**	**0.98**	**1882**	**29**	**1853**	**368**	**555**	**411**	**515**

Table 5. 3: Statistics at Vehicles

Statistics At vehicles	Total offload by Vehicles	Total executed at vehicle included locally executed and offloaded	Total Pending at Vehicles	Total Recv at Vehicles from other Vehicles and fog nodes	Total Task Gen by Vehicles	Avg waiting time at Vehicles	Offloaded tasks that are Computed at Vehicles	Direct delivery at Vehicles	One hop delivery at Vehicles	Two hop delivery at Vehicles	Failure delivery at Vehicles
Random Offloading	**24484**	**60159**	**6763**	**24484**	**86642**	**25.38**	**19034**	**17926**	**896**	**145**	**67**
Sequenced Offloading	**36414**	**49855**	**8454**	**36414**	**86784**	**24.46**	**28664**	**26640**	**1321**	**421**	**282**
Nearest Neighbour fog nodes	**29211**	**55196**	**6906**	**29211**	**86496**	**54.74**	**23138**	**21697**	**1033**	**267**	**141**
MAB Offloading with Random Waiting Time	**28410**	**55971**	**6826**	**28410**	**86653**	**20.39**	**22402**	**21315**	**833**	**191**	**63**
MAB Offloading with Selection Based on Waiting Time	**35876**	**50312**	**8384**	**35876**	**86784**	**24.56**	**28207**	**26212**	**1281**	**422**	**292**
MAB Offloading with Random Selection	**28846**	**55963**	**7186**	**28846**	**86691**	**20.6**	**22496**	**21209**	**983**	**219**	**85**

Table 5. 4: Total Statistics

Statistics At vehicles	Total Pending	Total Recv @ fog node	Total Recv @ Veh	Total Compute (Local + Offloaded)	Avg waiting time	Total Direct delivery	Total One hop delivery	Total Two hop delivery	Total Failure delivery	Average Efficiency	Total Pending
Random Offloading	**6892**	**1999**	**24484**	**62029**	**27.51**	**18277**	**1507**	**493**	**623**	**0.854**	**6892**
Sequenced Offloading	**8482**	**515**	**36414**	**50342**	**29.96**	**26656**	**1597**	**475**	**422**	**0.864**	**8482**
Nearest Neighbour fog nodes	**7206**	**2089**	**29211**	**56985**	**37.95**	**22037**	**1635**	**616**	**635**	**0.864**	**7206**
MAB Offloading with Random Waiting Time	**7228**	**2272**	**28410**	**57841**	**41.11**	**21704**	**1402**	**655**	**507**	**0.804**	**7228**
MAB Offloading with Selection Based on Waiting Time	**8412**	**596**	**35876**	**50880**	**25.81**	**26243**	**1574**	**497**	**457**	**0.868**	**8412**
MAB Offloading with Random Selection	**7215**	**1882**	**28846**	**57816**	**17.16**	**21577**	**1538**	**630**	**600**	**0.880**	**7215**

Waiting time of fog nodes

The waiting of fog nodes has shown the following behavior in all the corresponding variants of the algorithm.

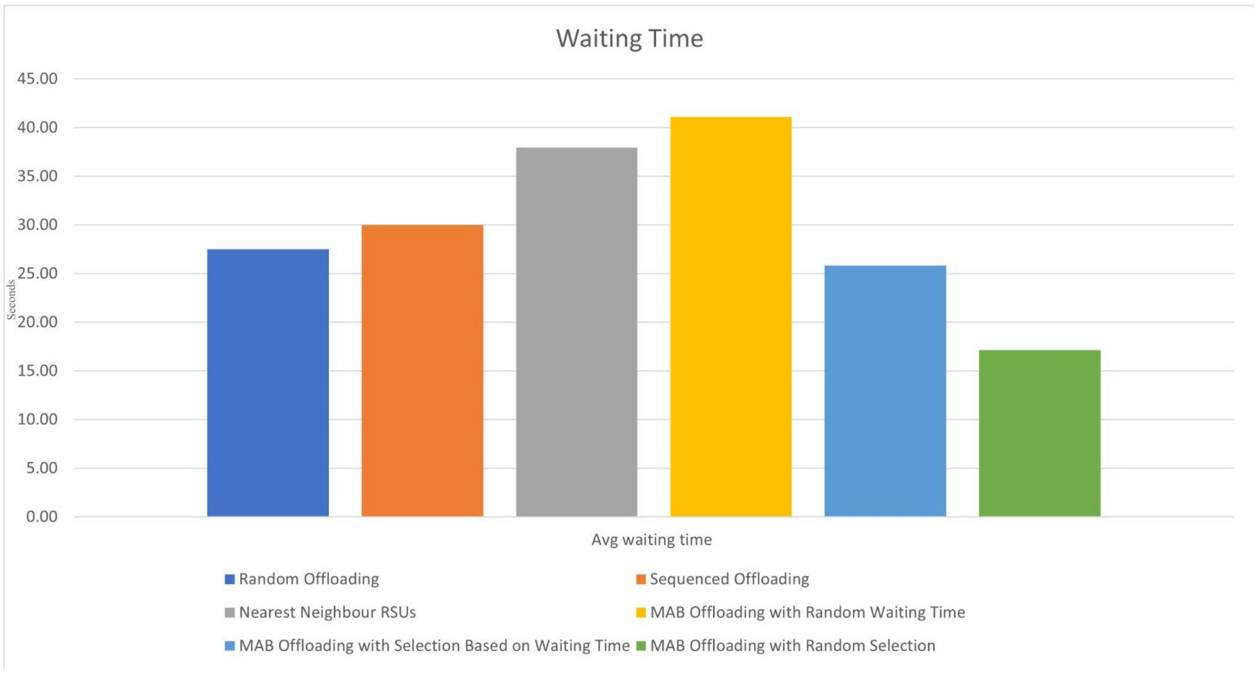


Figure 5. 1: Waiting Time at Fog Nodes

The figure shows that the lowest waiting time is generated by MAB offloading with random selection, while the MAB random waiting time has the highest waiting time after the nearest neighbor technique.

The two lowest variants for this parameter are MAB random and MAB selection based on waiting time.

Remember that one of the objectives here is to keep the waiting time minimum.

Efficiency at Fog Nodes

The efficiency of fog nodes has shown the following behavior in all the corresponding variants of the algorithm.

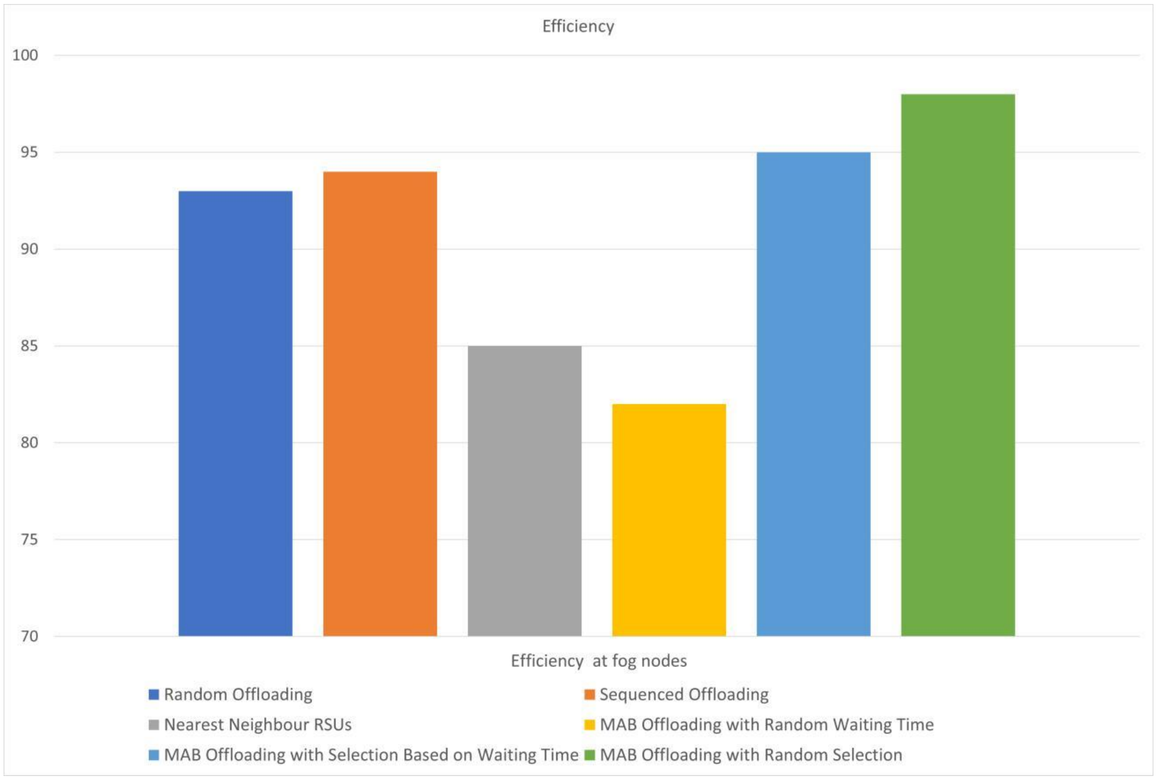


Figure 5. 2: Efficiency at Fog Nodes

The efficiency at fog nodes is the highest for the MAB random selection. While lowest for the MAB random waiting time. Although the random and sequenced offloading without MAB show good efficiency, it is lower than its corresponding MAB variants and they both have higher waiting times. So in the tradeoff for waiting for time and efficiency, the MAB random selection precedes while MAB selection based on waiting time follows it.

Waiting time at Vehicles

The waiting time of the vehicles is varied by the MAB algorithm even if it is being implemented at the fog nodes. Although this difference is not major slight variations are observed so they are included in this evaluation. The reason for this variation is that when the fog nodes are working efficiently in offloading the tasks and computing them well at a lower waiting time, the vehicles also perform their tasks faster as there is less waiting from the fog nodes and quicker response.

The waiting time variation is shown in the graph below,

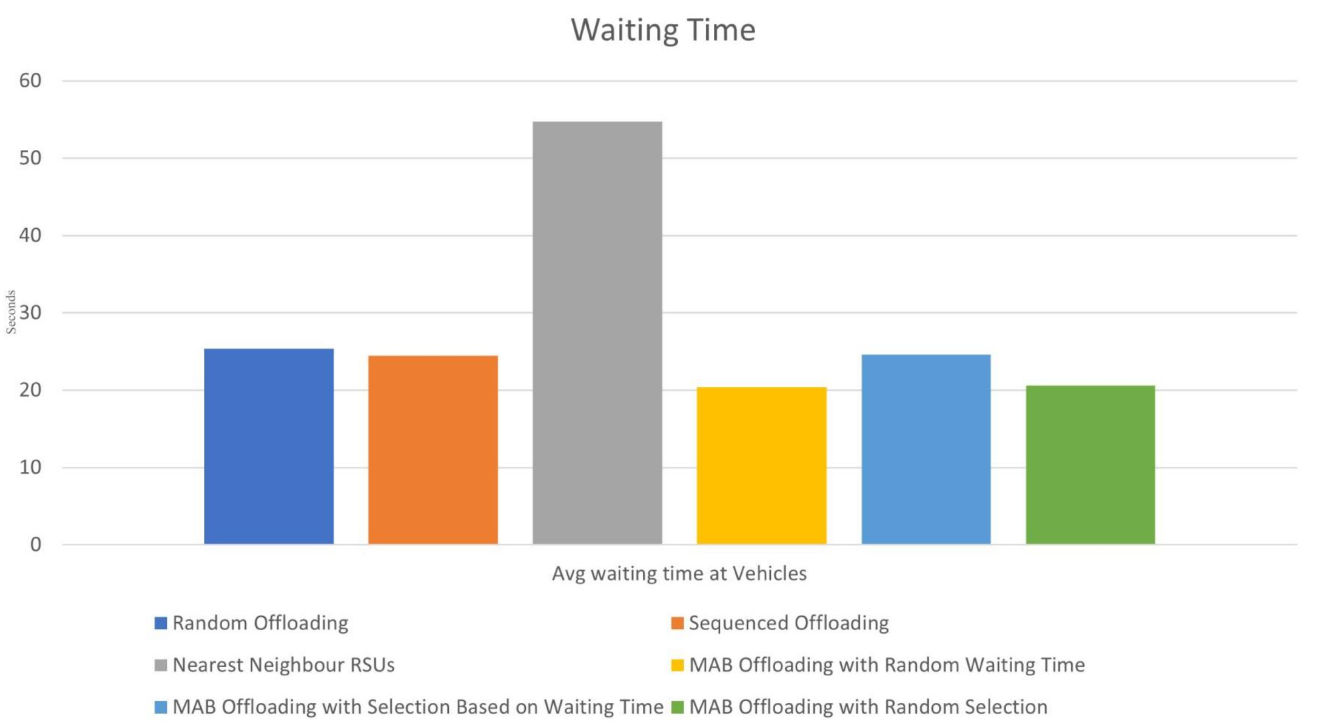


Figure 5. 3: Waiting Time at Vehicles

The vehicles show somewhat similar behavior as the fog nodes. The best or the least value for the waiting time is shown by the MAB offloading with a random selection scheme. While the most or worst value is displayed by the nearest neighbor technique. Although random waiting time is lying with the random selection, its reason is explained in the offloading statistics.

Efficiency at Vehicles

Similarly, the vehicles' efficiency also shows slight variation. The statistics of these variations are shown in the graph below,

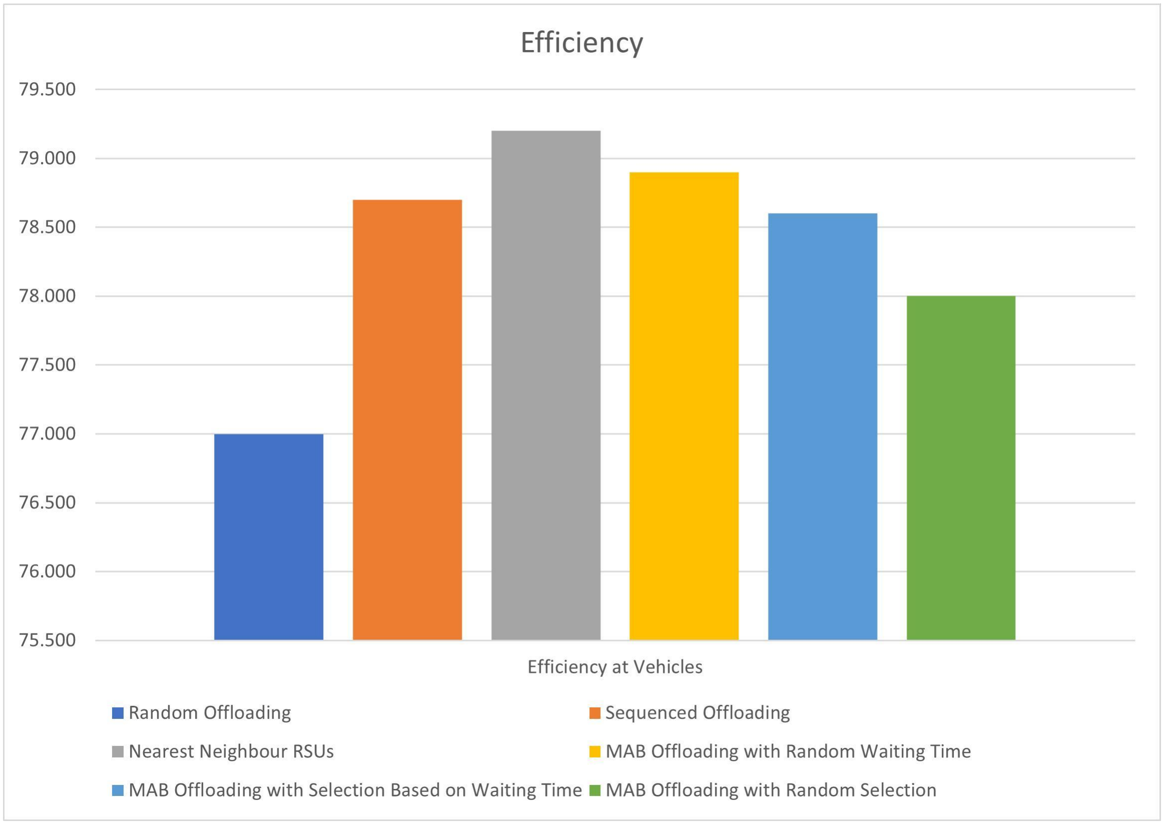


Figure 5. 4: Efficiency at Vehicles

The efficiency of vehicles is the highest in the nearest neighbor technique, however, the difference is quite low as the entire range of values inefficiency is between 77 and 79. The waiting time for the nearest neighbor is around 55 which is very high as compared to the corresponding variants. Hence, the efficiency of random selection and selection based on waiting time seems like the better choice among the rest.

Average Waiting Time

The average waiting time is calculated by taking the average of the waiting time of the vehicles and the fog nodes. These statistics are as follows,

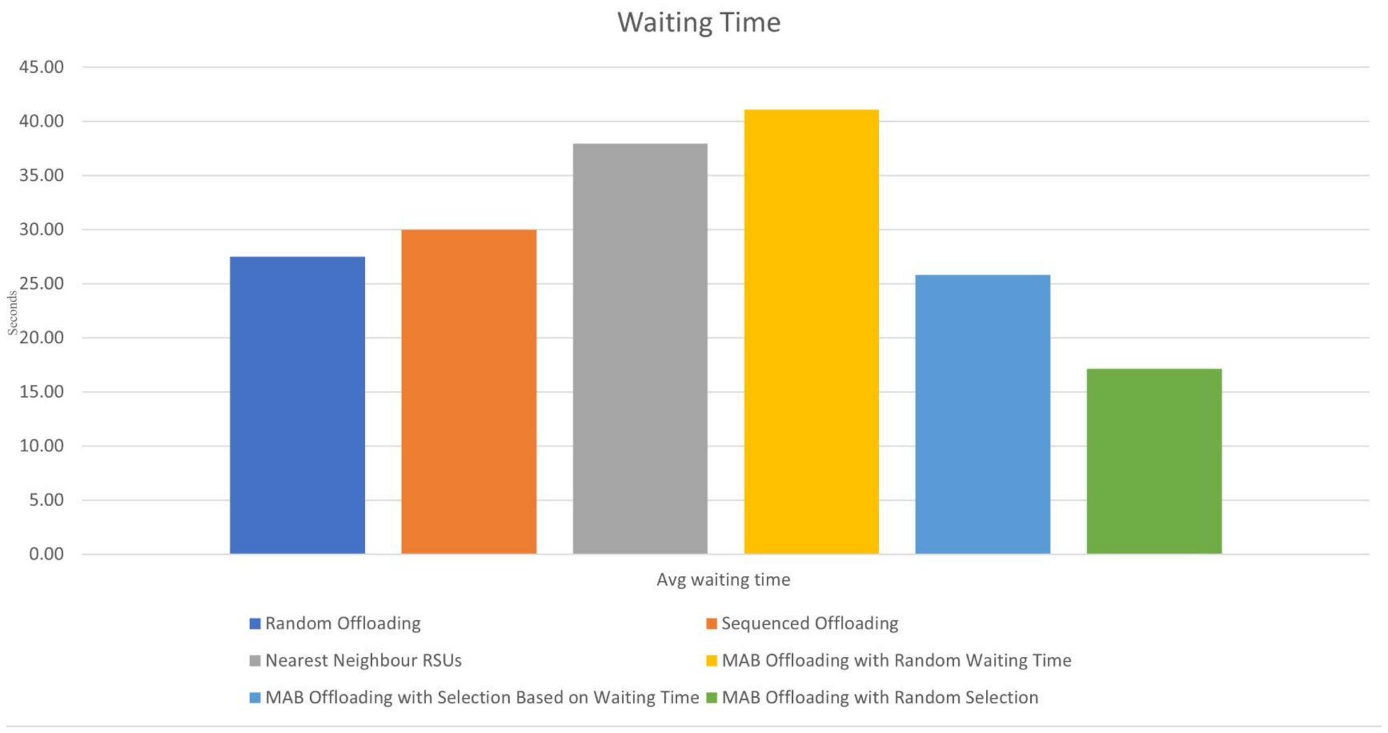


Figure 5. 5: Average Waiting Time

The lowest average waiting time is displayed by the MAB random reward allocation scheme. Followed by the MAB selection based on waiting time. The other techniques have performed well but MAB random and selection based on waiting time have outperformed the others.

Average Efficiency

The average efficiency of the simulation of the 6 techniques is given in the graph below.

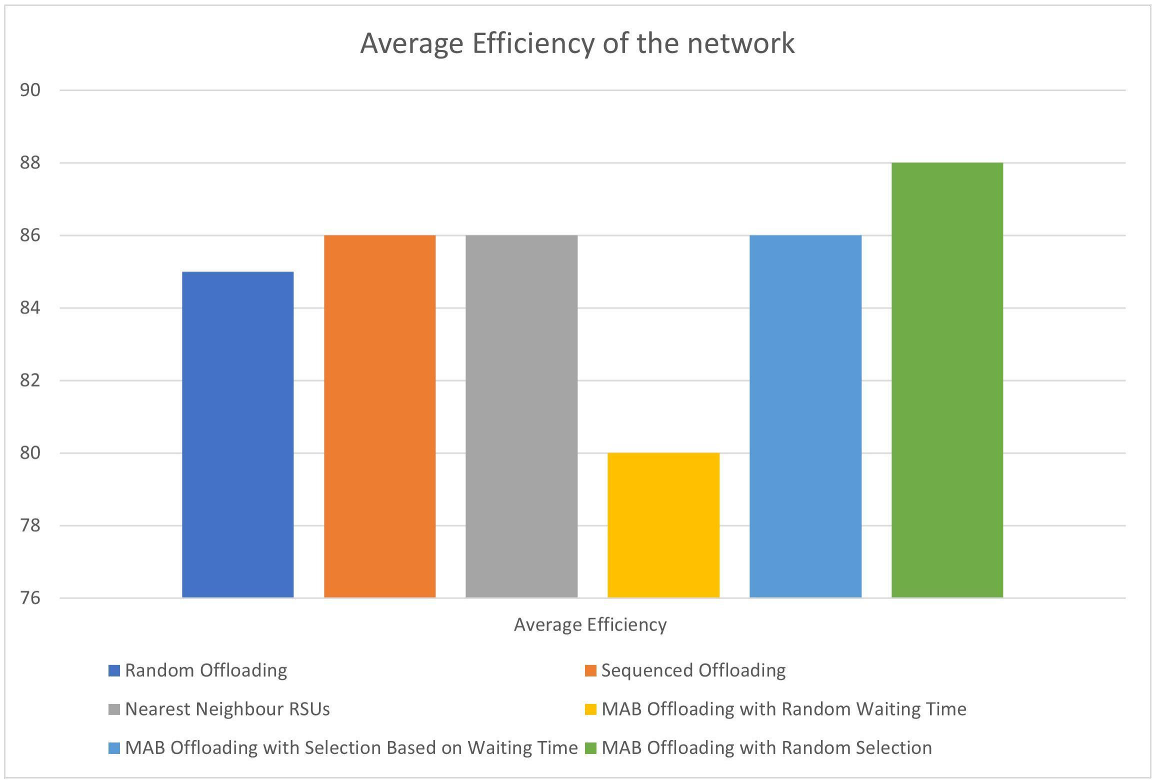


Figure 5. 6: Average Efficiency

Offloading at fog nodes

The offloading statistics are shown in the graph below

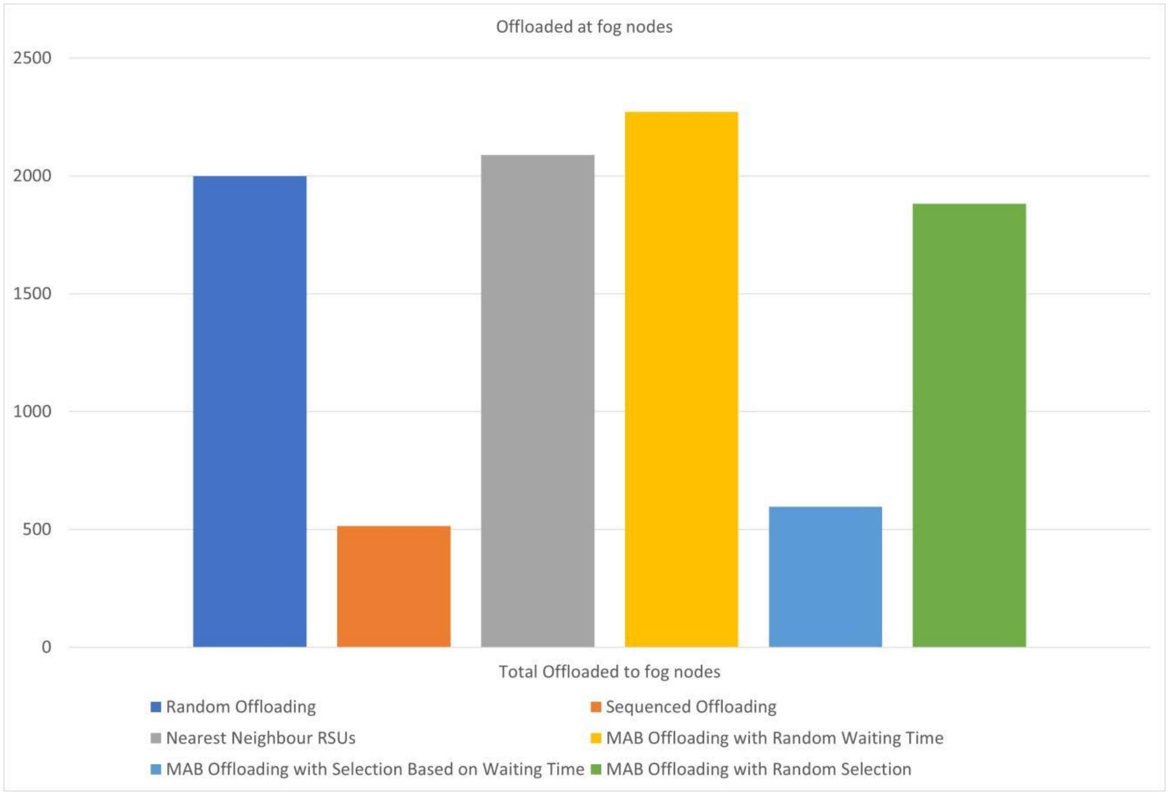


Figure 5. 7: Offloading at Fog Nodes

Although random offloading utilizes better resources than the MAB variants, the waiting time and efficiency are two important parameters that cannot be ignored. So considering the above statistics, it is quite clear in the above figures that the best tradeoff among the waiting time, efficiency, and resource utilization is shown in the two MAB variants; random selection and the selection based on waiting time.

Chapter 6

Conclusion, Limitations, and Future Work

Offloading and resource utilization are important problems, not only pertaining to the vehicular networks but all kinds of cloud and fog environments. Multiple resources are placed in the form of virtual or physical nodes around the smart cities in order to provide a functional network for all the entities in the city. These nodes contain a lot of resources that remain underutilized while tasks are being dropped or failed due to tasks waiting in the queue or excessive load on the task generating entity. In the vehicular network of a smart city, the tasks are mainly generated by vehicles. As the vehicle generates a task it is added to the queue for execution. If the task gets immediately executed, there is no need to offload it.

However, if the waiting time of the task exceeds a set threshold, the task needs to be offloaded to make space for new tasks in the queue as the vehicle is a real-time device that is continuously working and generating tasks. The question here is, where to offload this task? What can be the best decision for the vehicle that executes the task quickly and utilizes the resources available efficiently? Here, a deep learning-based reinforcement learning algorithm can come in handy. As the algorithm learns through every iteration what is the best behavior of the system that generates the best reward.

Multi-armed bandit are one such reinforcement learning algorithm that can be implemented in this scenario. The arms of the bandit are viewed as the available fog nodes for offloading. The number of arms and the fog nodes can be changed according to the algorithm or simulation's requirement. The algorithm assigns rewards to the arms or in this case the fog nodes, based on a parameter that is given by us. The parameter can be waiting time, execution time, distance of the fog node, or simply randomly assigned rewards that can be set for the algorithm.

The above-mentioned work has implemented the network through simulation in 6 different ways. A random approach of choosing fog nodes, a sequenced approach, and the nearest neighbor approach. It also implements a multi-armed bandit approach that assigns the rewards in a sequence, rewards are assigned to the neighboring nodes and the rewards are assigned randomly to each node while the algorithm learns the best node based on the reward they generate.

The first three approaches that do not contain the multi-armed bandit algorithm have shown results that have a higher waiting time and lower efficiency. Although the random approach has good efficiency the high waiting time on the fog nodes is a drawback in this approach.

The approaches that have implemented the MAB algorithm have shown that the algorithm generates better results i.e., a lower waiting time and a higher efficiency with promising offloading statistics as well. Among the approaches, the best results were seen in the multi-armed bandit random reward allocation approach. The lowest waiting was recorded for offloading through multi-armed bandit random reward allocation approach at 13.7 followed by 27.05 for MAB waiting for time-based reward allocation scheme. The other results and statistics show that the overall behavior of the multi-armed bandit random reward allocation approach is the most efficient among the other algorithms while all the algorithms show relatively good behavior.

Future Work:

There is further scope of research in this work by implementing the multi-armed bandit algorithm at the vehicle's end of the network. Right now, the algorithm is optimizing the offloading at the fog federation only. This can be extended to the entire network to observe how the network behaves if it is completely implemented based on the intelligent multi-armed bandit approach.

References

[1] C. Li, Y. C. Liu, and X. Yan, “Optimization-based resource allocation for software as a service application in cloud computing,” *J. Sched.*, vol. 20, no. 1, pp. 103–113, 2017.

[2] D. S. Linthicum, “Cloud computing changes data integration forever: What’s needed right now,” *IEEE Cloud Comput.*, vol. 4, no. 3, pp. 50–53, 2017.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Commun. Surv. Tutor.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] T. Alam, “IoT-Fog: A communication framework using blockchain in the internet of things,” *ArXiv Prepr. ArXiv190400226*, 2019.

[5] H.-J. Hong, “From cloud computing to fog computing: unleash the power of edge and end devices,” in *2017 IEEE international conference on cloud computing technology and science (CloudCom)*, 2017, pp. 331–334.

[6] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, “Vehicular fog computing: A viewpoint of vehicles as the infrastructures,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, 2016.

[7] Y. Toor, P. Muhlethaler, A. Laouiti, and A. De La Fortelle, “Vehicle ad hoc networks: Applications and related technical issues,” *IEEE Commun. Surv. Tutor.*, vol. 10, no. 3, pp. 74–88, 2008.

[8] P. Papadimitratos, A. De La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, “Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation,” *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 84–95, 2009.

[9] Y. Wu, J. Wu, G. Zhou, and L. Chen, “A direction-based vehicular network model in vehicular fog computing,” in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 585–589.

[10] F. J. Ros, J. A. Martinez, and P. M. Ruiz, “A survey on modeling and simulation of vehicular networks: Communications, mobility, and tools,” *Comput. Commun.*, vol. 43, pp. 1–15, 2014.

[11] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, “A comprehensive survey on vehicular ad hoc network,” *J. Netw. Comput. Appl.*, vol. 37, pp. 380–392, 2014.

[12] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, “Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges,” *IEEE Commun. Surv. Tutor.*, vol. 16, no. 1, pp. 337–368, 2013.

[13] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, “A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing,” *IEEE Commun. Surv. Tutor.*, vol. 15, no. 3, pp. 1294–1313, 2012.

[14] I. Manickam, A. S. Lan, and R. G. Baraniuk, “Contextual multi-armed bandit algorithms for personalized learning action selection,” in *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017, pp. 6344–6348.

[15] T. L. Lai and H. Robbins, “Asymptotically efficient adaptive allocation rules,” *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.

[16] S. S. Villar, J. Bowden, and J. Wason, “Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges,” *Stat. Sci. Rev. J. Inst. Math. Stat.*, vol. 30, no. 2, p. 199, 2015.

[17] C. Tekin and E. Turgay, “Multi-objective contextual multi-armed bandit problem with a dominant objective,” *ArXiv Prepr. ArXiv170805655*, 2017.

[18] J. Nino-Mora, “Stochastic Scheduling.,” *Encycl. Optim.*, vol. 5, pp. 367–372, 2009.

[19] J. White, *Bandit algorithms for website optimization*. O’Reilly Media, Inc., 2012.

[20] R. Sen, K. Shanmugam, A. G. Dimakis, and S. Shakkottai, “Identifying best interventions through online importance sampling,” in *International Conference on Machine Learning*, 2017, pp. 3057–3066.

[21] K. Su, J. Li, and H. Fu, “Smart city and the applications,” in *2011 international conference on electronics, communications, and control (ICECC)*, 2011, pp. 1028–1031.

[22] F. Wortmann and K. Flüchter, “Internet of things,” *Bus. Inf. Syst. Eng.*, vol. 57, no. 3, pp. 221–224, 2015.

[23] A. Al Mamun, G. Guo, and C. Bi, *Hard disk drive: mechatronics and control*. CRC press, 2017.

[24] M. Cornwell, “Anatomy of a solid-state drive,” *Commun. ACM*, vol. 55, no. 12, pp. 59–63, 2012.

[25] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, “Cloud storage as the infrastructure of cloud computing,” in *2010 International Conference on Intelligent Computing and Cognitive Informatics*, 2010, pp. 380–383.

[26] B. Ahmed, A. W. Malik, T. Hafeez, and N. Ahmed, “Services and simulation frameworks for vehicular cloud computing: a contemporary survey,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2019, no. 1, pp. 1–21, 2019.

[27] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, “Connected vehicles: Solutions and challenges,” *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, 2014.

[28] X. Lyu, H. Tian, C. Sengul, and P. Zhang, “Multiuser joint task offloading and resource optimization in proximate clouds,” *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, 2016.

[29] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, “Energy-optimal mobile cloud computing under stochastic wireless channel,” *IEEE Trans. Wirel. Commun.*, vol. 12, no. 9, pp. 4569–4581, 2013.

[30] W. Zhang, Y. Wen, and D. O. Wu, “Energy-efficient scheduling policy for collaborative execution in mobile cloud computing,” in *2013 Proceedings Ieee Infocom*, 2013, pp. 190–194.

[31] W. Zhang, Y. Wen, and D. O. Wu, “Collaborative task execution in mobile cloud computing under a stochastic wireless channel,” *IEEE Trans. Wirel. Commun.*, vol. 14, no. 1, pp. 81–93, 2014.

[32] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, “Computation Offloading in Heterogeneous Vehicular Edge Networks: On-line and Off-policy Bandit Solutions,” *IEEE Trans. Mob. Comput.*, 2021.

[33] X. Wang, X. Wei, and L. Wang, “A deep learning based energy-efficient computational offloading method in Internet of Vehicles,” *China Commun.*, vol. 16, no. 3, pp. 81–91, 2019.

[34] A.-M. Ahmad, H. Idriss, A. El Mouallem, and Z. El Bazzal, “Chain-based data dissemination in vehicular ad-hoc networks (VANETs),” in *2018 Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*, 2018, pp. 75–80.

[35] S. S. Shah, A. W. Malik, A. U. Rahman, S. Iqbal, and S. U. Khan, “Time barrier-based emergency message dissemination in vehicular ad-hoc networks,” *IEEE Access*, vol. 7, pp. 16494–16503, 2019.

[36] Z. Ning *et al.*, “A cooperative quality-aware service access system for social Internet of vehicles,” *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2506–2517, 2017.

[37] L. Guo, Z. Ning, W. Hou, B. Hu, and P. Guo, “Quick answer for big data in sharing economy: Innovative computer architecture design facilitating optimal service-demand matching,” *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1494–1506, 2018.

[38] W. Hou, Z. Ning, L. Guo, and X. Zhang, “Temporal, functional and spatial big data computing framework for large-scale smart grid,” *IEEE Trans. Emerg. Top. Comput.*, vol. 7, no. 3, pp. 369–379, 2017.

[39] F. Jiang, W. Liu, J. Wang, and X. Liu, “Q-Learning Based Task Offloading and Resource Allocation Scheme for Internet of Vehicles,” in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, 2020, pp. 460–465.

[40] Y. Saleem, N. Mitton, and V. Loscrì, “A Vehicle-to-Infrastructure Data Offloading Scheme for Vehicular Networks with QoS Provisioning,” 2021.

[41] M. Lee and C. S. Hong, “Service Chaining Offloading Decision in the EdgeAI: A Deep Reinforcement Learning Approach,” in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2020, pp. 393–396.

[42] S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, “Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks,” *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 49–55, 2019.

[43] X. Huang, K. Xu, C. Lai, Q. Chen, and J. Zhang, “Energy-efficient offloading decision-making for mobile edge computing in vehicular networks,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2020, no. 1, pp. 1–16, 2020.

[44] J. Wang, T. Lv, P. Huang, and P. T. Mathiopoulos, “Mobility-aware partial computation offloading in vehicular networks: A deep reinforcement learning based scheme,” *China Commun.*, vol. 17, no. 10, pp. 31–49, 2020.

[45] M. Ashritha and C. S. Sridhar, “RSU based efficient vehicle authentication mechanism for VANETs,” in *2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO)*, 2015, pp. 1–5.