AMERICAN UNIVERSITY OF BEIRUT

An Opportunistic Vehicle Based Computing for IoT Offloading

by

Khaled Sarieddine

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Science
to the Department of Computer Science
of the Faculty of Arts and Sciences
at the American University of Beirut

Beirut, Lebanon
September 2020

AMERICAN UNIVERSITY OF BEIRUT

An Opportunistic Vehicle Based Computing for IoT Offloading

by

Khaled Sarieddine

Approved by:

Haidar Safa [illegible]

Dr. Hiadar Safa, Professor — Advisor

Computer Science

[illegible]

Dr. Hassan Artail, Professor — Member of Committee

Electrical and Computer Engineering

[illegible]

Dr. Wassim El Hajj, Professor — Member of Committee

Computer Science

Date of thesis defense: August 20, 2020

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Sarieddine Khaled Salah

Last	First	Middle
Sarieddine	Khaled	Salah

[✓ Master's Thesis] ○ Master's Project ○ Doctoral Dissertation

☐ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☒ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: [✓ One] ___ year from the date of submission of my thesis, dissertation or project.
Two ___ years from the date of submission of my thesis, dissertation or project.
Three ___ years from the date of submission of my thesis, dissertation or project.

Khaled
Signature

17-9-2020
Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

I would like to dedicate a special thanks to my advisor Professor Haidar Safa and and co-advisor Professor Hassan Artail for their time, supervision, support, patience, motivation, enthusiasm, immense knowledge, and invaluable criticism throughout my thesis work. I would also like to thank the rest of the thesis committee, Professor Wassim El Hajj, for his encouragement, insightful comments, and time.

Dedication

First, I would like to thank my family for the encouragement and support that they have provided me with, continuously and unconditionally. Thank you for my second home.

Thank you to all my friends, and especially my favorite most fantastic boo, for the stimulating discussions, the sleepless nights working together before deadlines, and for all the fun that we have had in the last two years.

Finally my sincere gratitude goes to the Faculty of Arts and Sciences Dean's office, who sponsored my masters education, and especially to Mrs. Zeina Halabieh.

An Abstract of the Thesis of

Khaled Sarieddine for Master of Science
Major: Computer Science

Title: An Opportunistic Vehicle Based computing for IoT offloading

IoT is one of the most prolific origins of data that is collected from sensory inputs. However, IoTs are characterized by low computational power, thus motivating the data-offloading scheme, a promising technique that improves energy performance issues in limited power devices. Instead of offloading to fog or cloud, a new research track is emerging where devices unload their data to vehicles roaming around, and this particular type of offloading is called vehicular cloud offloading. In this thesis, we propose the use of vehicles as Mobile Computing Nodes roaming around the cities to offer computational services to IoT devices. Choosing the most appropriate MCN by an IoT device that wishes to offload specific communication tasks to it will be formulated as an optimization problem. Indeed, we choose the top N MCNs within the range of the fixed IoT according to the SINR and the time of connection, taking into consideration the mobility of the MCN. Another challenge would be delivering the results back to the IoT device or another party, knowing that the vehicles will not be stationary.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The revolutionary changes in the automotive industry lead companies to invest in the incorporation of advanced technological features in smart vehicles. Modern cars like Tesla, are equipped with powerful onboard computers, storage devices, sensitive radio transceivers, collision radars, and GPS devices. These smart vehicles are considered as computers-on-wheels. Although cloud and fog computing are advancing, there are some issues to adopting these two paradigms in rural areas and in hazardous and volatile situations that motivate the transition to mobile vehicular clouds where the vehicles can service computation requests by various external devices and replacing the traditional paradigms [13].

Rural areas suffer from inherently limited internet and cellular coverage. Moreover, these areas, like other urban ones, need monitoring for bridges' health, earthquakes, temperature, humidity. These conditions, among others, are monitored using low computation ability and limited power devices, which are called IoTs. For these devices to handle all the data it collects that need computation, they adopt the offloading paradigm where they migrate their data computation to the nearby computation resource provider, which could be cloud, fog, or in this case to vehicular cloud where IoTs will opportunistically take advantage of passing by vehicles. In this chapter we discuss the motivation behind our work, along with defining our problem, listing our objectives and contribution and finally detailing the thesis organization.

1.1 Motivation

The motivation for an opportunistic vehicle based computing architecture to service IoT applications lies in the unique advantages of enabling IoT services in hazardous and volatile situations, where resources and connectivity are inherently limited and cannot be sufficiently supported by static fog or cloud services.

Clouds are distant and suffer from unreliable long latencies over WANs, which is already recognized as an impediment for performing resource and data-intensive computations with cloud computing systems. Moreover, the energy consumption of data centers is a daunting problem that has gained significant attention from the research community. Although placing serving resources at the edge of the internet, which is the core idea behind several solutions like cloudlets or fog nodes, might address the issues mentioned above, it suffers from limited computation ability and similar to the cloud's lack of mobility. Installing fog nodes is quite challenging, and the nodes that are installed might get overwhelmed by the amount of data it receives. Thus, motivating the vehicular cloud concept! Where VANETs can act as mobile fog nodes offering services as they move around the city, thus increasing the availability of resources. This solution is a cost-effective resource and makes use of idle resources roaming around without additional cost for installation. Briefly, the motivation behind our work also, is the under-utilization of vehicular resources, and decrease the cost for the service providers to use existing resources.

1.2 Problem Definition

Various issues are faced by traditional fog and cloud computing paradigms. These issues can be summarized as following:

- Costs to scale up and to install new fog nodes are high compared to the vehicular cloud, which uses already existing idle resources.
- Energy consumption by data centers is significant.
- Lack of mobility of the cloud and fog nodes makes it impossible to service remote areas where the vehicular cloud fits better.
- The limited computation ability of fog nodes might make them susceptible to being overwhelmed by the amount of data it receives.

These issues faced by traditional paradigms motivates the transition to vehicular clouds. Indeed, vehicles are being equipped with powerful computers, roaming around the cities carrying idle resources yet to be utilized, it was proved through previous research as a great alternative to the traditional offloading and computational paradigms.

1.3 Objectives and Contribution

Although previous research has been done in this domain, we will discuss the proposed approaches focusing on their limitations. This thesis aims to introduce a new delay-tolerant communication protocol. It simplifies the offloading mechanism from IoT devices to vehicles directly without the overhead of having a resource manager, or any intermediate subsidiary while taking into consideration the mobility of the car and the Signal Interference Noise Ratio (SINR) as a selection criterion. To this end, we frame the contributions of this thesis as follows:

- Survey traditional and modern offloading paradigms and identify their limitations.
- Propose an opportunistic vehicle based computing framework for IoT offloading. A framework that defines the means of communication between IoT and a mobile vehicular node in a heterogeneous environment, which allows the IoT to offload computation to a nearby vehicle and retrieve the result back.
- Implement the propose approach using ns3 as a network simulator and SUMO as a traffic simulator.
- Evaluate the performance of our proposed framework.

1.4 Thesis organization

The remainder of this thesis is organized as follows. In chapter 2, we include some background and basic concepts related to our proposed approach. In chapter 3, we present a number of related work in the context of VANET offloading schemes. In chapter 4, we present our proposed approach. In chapter 5, we perform thorough empirical evaluations of the proposed approach and report the results. Finally, we conclude in chapter 6 and pinpoint a few research endeavors for future work.

Chapter 2

Basic Concepts and Background

Devices surrounding us such as sensors, mobile phones, cars, and smart homes cannot communicate without the existence of proper communication protocols, infrastructure, and third parties. Sensors, which are computational and energy limited devices, need some offloading mechanisms to analyze all the data they acquire from their environment. This chapter provides some basic concepts of the components which usually interact with each other to form the interconnected world around us, exploring the traditional offloading mechanisms and architectures and explaining how applications interact with each other. Finally, listing application areas.

2.1 Internet of Things

The Internet of Things (IoT) is a system of interrelated devices characterized by low computing power and the ability to send data over a network. An IoT network consists of web-enabled smart devices that acquire data from their environment using embedded processors, sensors, and communication hardware to collect, send, and act on it. The devices send the collected data to the IoT gateway or other edge devices where data is sent to the cloud to be analyzed; however, IoT devices are characterized by low computing capabilities and limited power supply [14]. Hence, these limitations make the notion of offloading processing tasks to more powerful nodes inevitable. IoT made its way to become an asset in many industries contributing to revolutionizing these industries, such as agriculture, food processing, environmental monitoring, security surveillance, and others. Meanwhile, the number of IoT applications is rapidly growing [15].

2.2 Cloud Computing

Cloud computing relies not only on sharing but also on maximizing resources, which are critical requirements for the IoT platform. It offers elasticity and scalability of resources and applications [16].The users can access the cloud services from any location and with any device through the internet, which characterizes the cloud as location independent. An essential component for the offloading mechanism is an efficient task distributor among servers such as AutoScaler which is proposed by [17] where they investigated the effect of large-scale offloading for IoT. The constant increase in demand for computation leads to a significant improvement over the years in the computation-offloading paradigm. The processors' capability is limited due to the limited power supply, which remains a critical limiting factor in the design of mobile applications [18]. Mainly cloud offloading frameworks follow three steps: application partitioning, preparation, and offloading decision. The main motivation of cloud offloading is assigning intensive components to cloud server and relief edge devices from the burden of computing themselves such as CloneCloud [19], which is a framework that aims at improving the battery life and performance on the mobile device. Similarly, [20] and [21] indicate the main objective of cloud computing, which is to enhance user performance and save energy on edge devices.

Cloud computing is one of the first paradigms adopted for offloading. This paradigm depends on internet connection and network bandwidth to perform offloading from the limited power devices to the cloud. However, this type of communication creates a bottleneck since, as the number of devices requesting computation increases, the network becomes more congested, which leads to delays in the transfer of data to and from the cloud. Since some of this computation sometimes are time-sensitive thus, it cannot tolerate any delays; researchers suggested to bring the computation to the edge of the network and introduced fog computing.

2.3 Fog Computing

Fog Computing refers to bringing cloud intelligence near the edge. It facilitates the operation of computation and networking services between end devices (IoT) and cloud computing data centers. Unreliable latency, lack of mobility support, and location-awareness are issues induced by cloud computing. The elasticity of the resources and services provided by fog nodes can address the above challenges. Fog computing complements the cloud computing paradigm from the core to the edge of the network [22]. Introducing fog nodes to the IoT networks would resolve the transmission and offloading issues that IoT networks suffer from when communicating with the cloud [23]. Due to the delays introduced by cloud computing, fog computing became a better approach to facilitate the operation of computation and networking services [22] [23] [24]. IoT needs a platform that supports rapid mobility patterns, even requiring, in some cases, high throughput on demand. Also, it must support systems requiring reliable sensing, analysis, control, and actuation, in scenarios subject to reduced or unreliable connectivity or requiring very low latency. It should also be able to manage a large amount of geographically distributed things that may produce data that require different levels of real-time analytics and data aggregation [24]. Thus, the cloud faces a challenge when it comes to the previously mentioned platform requirements. However, these challenges are addressed with fog computing, which makes it a natural platform for IoT. However, there will always be an interplay between fog and cloud computing since they complement each other.

The main consequence of having a cloud server between the end device and the controller taking into consideration jitters and delays is that they are able to apply mitigation mechanisms to deal with the delays and jitters caused by the networks when the controller is offloaded to the cloud or fog. Moving the controller to a remote server degrades the server performance [25]. However, the the performance shows improvement when it comes to fog since it is closer to the end devices than the cloud. In the era of big data sending enormous amounts of data may be inefficient due to the high cost of communication bandwidth and high redundancy of data (for example, periodic sensor readings).

Moreover, fog nodes incorporate a delay-minimizing policy whose goal is to reduce service delay for IoT nodes [23], utilizing IoT-fog, fog-cloud, and fog-fog interaction to reduce the lag by sharing some of the load. The decision to offload a task is based on the response time of a fog node, which depends on several factors, such as the amount of computation needed, queuing status (current load), and the processing capabilities of a fog node. Fog nodes collaborate to fulfill the requests sent from IoT nodes to the fog layer. However, if the fog node receiving requests is busy, it will offload some of the computation to its neighboring fog nodes. In fog computing, they use distributed mode of interaction, where there is no central node; instead, fog nodes in a domain run a protocol to distribute their state information to the neighboring nodes in the same domain. Moreover, requests are put in a priority queue so that they obtain a fair scheduling mechanism.

The number of connected devices to the cloud increased tremendously in the last ten years, and large amounts of data are being produced. Offloading data and computation to the cloud became the mainstream. However, data privacy becomes an essential issue as IoT devices send all their data to the cloud. Data synchronization is an important feature fog based computing [26]. By offloading some of the computation to the fog servers, privacy can be guaranteed. In addition, a differential synchronization algorithm decreases the communication cost. Its emergence reduces the volume of uploaded data to the cloud by only changing what is required and then extended the method by introducing a Reed-Solomon code for security consideration. For example, if a user changed a couple of blocks frequently traditional differential would sync every time the user updates something however in this approach the differential will only sync after the users amount of data exceeds a threshold on the fog node, only then the fog nodes will sync to the cloud. Thus, the fog nodes act as relaying nodes.

Fog nodes are characterized by being stationary, which is considered as a limitation in some scenarios such as natural disasters where the communication infrastructure would fail to serve its purpose. Thus, comes the concept of vehicular fog nodes, IoTs can take advantage of the roaming mobile cars [27] [28].

2.4 Vehicular Ad hoc Networks

With vehicular manufacturing advancement, the capabilities of vehicles are increasing, whereby they can provide computational services, and can also communicate with external devices. Recently, the research community suggested that vehicles could form vehicular mobile clouds where cars can be used as a resource not only for data relaying, infotainment, and sensing but also for data storage and computing for efficient utilization of available resources in neighboring vehicles [29] [2]. However, due to sizeable sensory data inputs, strict latency requirements, and dynamic wireless networking conditions, offloading vehicular applications to the cloud is very challenging [30]. These smart vehicles are being equipped with a powerful on-board unit (OBU), trusted platform module (TPM), and sensors. OBUs, which are computers, designed to handle vehicle-to-vehicle (V2V) and vehicle to infrastructure (V2I) communication and run programs required by the vehicle and the user. The on-road infrastructure communication units are called roadside unit (RSU), which are equipped with powerful computing devices and installed in different locations around the city.

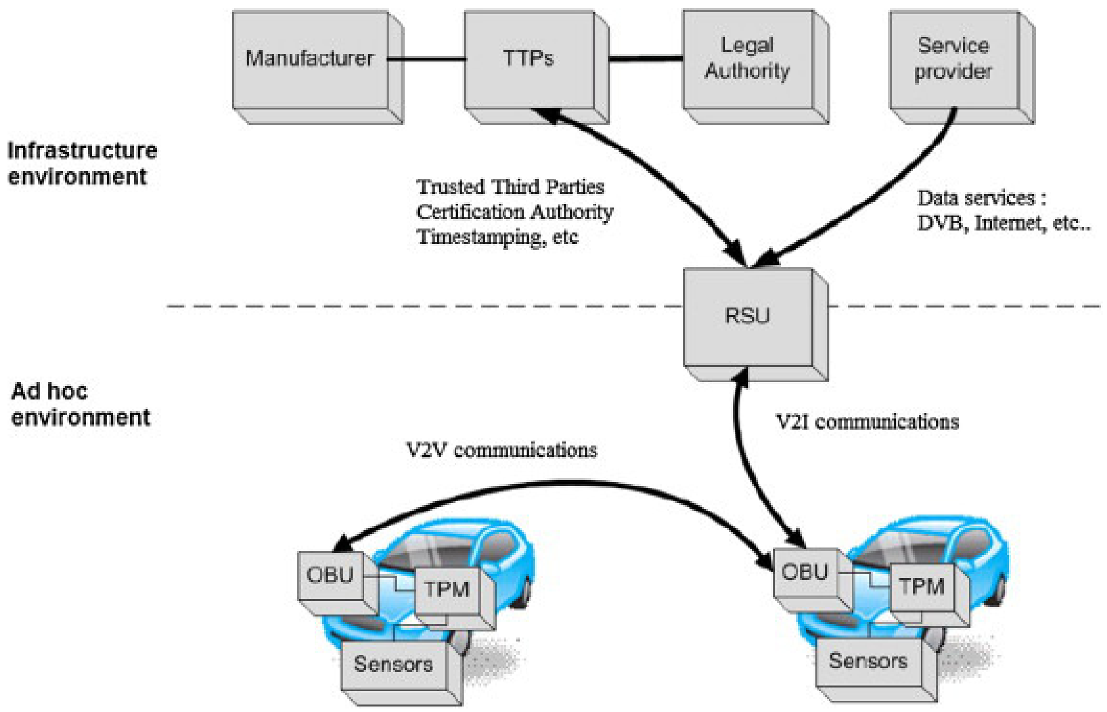


Figure 2.1: Vehicular Ad hoc Networks components

The vehicles are equipped with GPS and Wi-Fi devices that enable vehicle to vehicle (V2V) communication forming a vehicular ad-hoc network (VANET). Real-time peer to peer vehicle communication is necessary to share up-to-date information about road and traffic conditions and provide essential services such as rerouting traffic effectively through dense urban areas, prevent collisions and avoid car accidents [31] [32] [33]. In addition to V2V, VANETs compromise vehicle to infrastructure (V2I) communications, which offer direct communication between vehicles to and from RSUs, which are a prerequisite for VANETs communication. In this mode, for a car to connect and communicate with external networks, such as the internet, it needs to establish a connection with the RSU [34]. OBU, TPM, and sensors constitute the Ad hoc part of the environment, whereas the infrastructure environment includes manufacturer, trusted third party units (TTP), service providers, and legal authorities. RSU acts as a bridge between Ad hoc and infrastructure parts. Figure 2.1 shows the infrastructure and Ad hoc environments, which form a simplified VANET network. V2V and V2I communication are coupled with dedicated short-range communication (DSRC) protocol since vehicular safety communication cannot tolerate long establishment delays before being enabled to communicate with other vehicles encountered on the road or to the RSUs. IEEE 802.11p standard for the DSRC reduces the connection setup overhead and suites vehicular communication well [35].

2.5 Offloading Paradigm

Data, which is now one of the most valuable merchandise in the world, is increasing tremendously. IoT is one of the most prolific origins of data that is collected from sensory inputs. These IoTs, as mentioned earlier, are characterized by low computational power, which means there is no way that they can handle all this data, thus motivating the offloading scheme of data in order to enhance the IoT environment and lessen the burden of computation upon them.

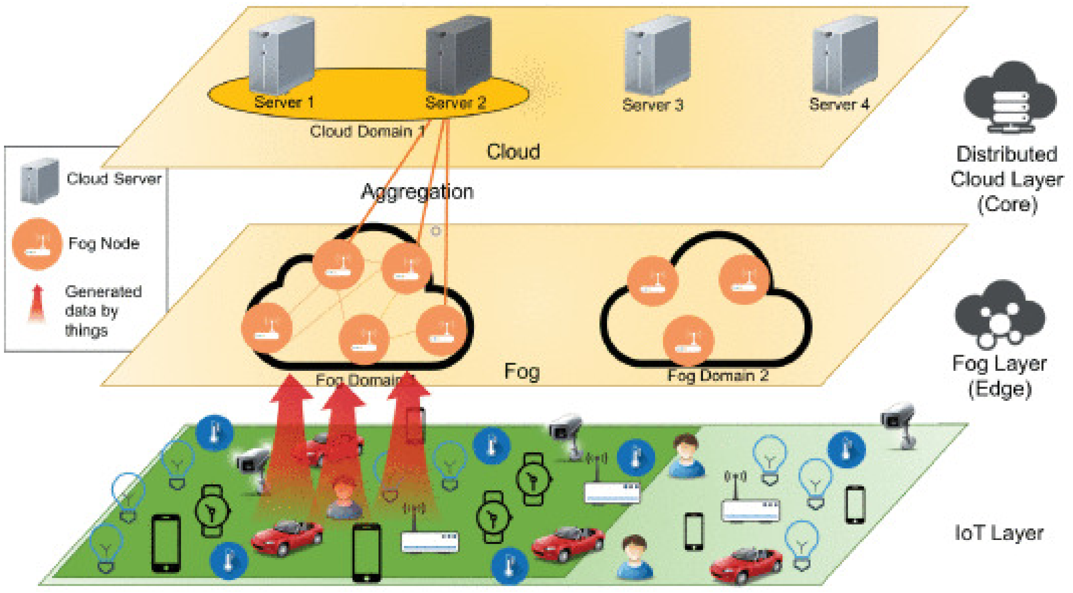


Figure 2.2: General IoT-fog-cloud framework

Offloading is a promising technique that improves energy and performance issues in limited power devices. A task is opportunistically outsourced from a device when in the presence of network connectivity, the device can reach the server at relatively low latency rates of transfer [17]. However, sometimes offloading is not beneficial if the computational requirements are small compared to communication costs (latency, energy). Thus, offloading should only occur when its benefits are significant compared to the cost of running the task on the device. The ultimate goal is to extend the battery life by reducing the overall amount of processing by a device [36] [37]. Many types of offloading have been suggested and applied. First, cloud computing, where IoT or any computationally limited device offloads data through the internet to cloud servers that are handled by third parties, second, offloading to Fog nodes, which are nodes that bring the computation to the edge of the network by providing services that are usually offered by the cloud servers to the IoT environment. Finally, vehicular cloud offloading where IoT offloads its computation to vehicles. Generally, the overall architecture looks like the one in Figure 2.2 where the IoT layer, which constitutes of variety of IoT devices, communicates with the fog nodes layer (clusters of fog nodes) that, in turn, interacts with the distributed cloud layer [23], showing how all the above components interact with each other in a complementary manner.

2.6 Application Areas

The diversity of the applications that can use our approach embodies a great potential; it can be incorporated in all areas of every-day life of individuals, enterprises, and societies. IoT application covers smart environments in various domains such as transportation, building, city, lifestyle, retail, agriculture, factory, supply chain, emergency, healthcare, user interaction, culture and tourism, environment, and energy [38] [39] [40].

2.6.1 Smart Cities

Concrete, like any other commodity, has a service life. IoT can be used to report data about the structure of health by monitoring the vibrations and material conditions in buildings, bridges, and historical monuments. For example, the structural health of bridges is a vital and critical issue that requires fast analysis of the data reported by the sensors and IoTs that are put in order to monitor the conditions in bridges. Thus, instead of sending these data to the cloud, it will opportunistically take advantage of the vehicles passing on them to perform the needed computation in a timely manner.

Moreover, intelligent and adaptive weather streetlights the sensors send weather-related data to our MCNs, which would analyze the received data and return the action that should be done by the traffic system to adapt to the weather conditions sent by the sensors. Not only in intelligent smart lightning, it can be used, but also, in intelligent transportation, where sensors might send data about weather conditions and traffic, our MCNs will be able to find the best routes to get past the traffic and unexpected events like accidents or analyze the weather data input. In addition, it can help in digital video monitoring, fire control management. Where the sensors take advantage of the passing vehicles nearby to analyze the data regarding these issues, instead of sending a video feed to the cloud to be analyzed, it can be processed by casually passing by vehicles. Likewise, detecting the rubbish levels in garbage containers through sensors placed in the containers which can send its data to passing by vehicle to analyze the levels and report back to the responsible entity the result of this computation. Thus, helping in optimizing the routes for garbage collection.

2.6.2 Smart environments

In the smart environment, the weather conditions such as humidity, temperature, pressure, wind speed, and rain are periodically monitored such data needs to be analyzed, similarly reporting plates vibrations to ensure early earthquake detection. On top of that, measuring the levels of waters in rivers, dams, and reservoirs during rainy days or the quality of water, these tasks cannot be performed by simple sensors. Thus, they need to offload their collected data to another party to do such computation on their behalf.

2.6.3 Smart industries

The detection of gas levels and leakages in industrial environments is critical and time dependent. Where some MCNs passing by a factory can analyze the reported data and reply with what do these data imply. The low latency ensured by adopting the MCNs will ensure the safety of workers in a chemical plant where some sensors might be reporting oxygen levels.

2.6.4 Smart health systems

In hospitals, the life of an endangered patient can be saved if the critical moment was reported as soon as possible. Thus, monitoring the conditions of patients in a hospital and in old peoples home is necessary, it can help save lives. Therefore, these monitoring sensors can offload their data to the passing by MCNs that will analyze the data and based on its response; it can help save lives due to its quick response and its occasional availability.

2.6.5 Smart Energy

Smart grids are a potential customer for our approach by analyzing the monitored energy consumption. Along with that, analyzing the flow of energy from wind turbines, powerhouse, and two-way communication with customers smart meters to analyze consumption patterns.

2.6.6 Real-life Scenarios

There are different types of sensors found, some measure temperature; others measure humidity, wind speed, solar radiation, or barometric pressure. All these sensors can be installed throughout forests in order to monitor and predict the breakout of wildfires like those that happened a while ago in Australia. All the data gathered through these sensors need to be analyzed and undergo some machine learning models where these different types of data are inputted as various features to the machine learning model and result in a prediction if there would be any fire. This result if it indicated the possibility of fire it will return the result to the IoT, which would initiate an alert to the specific authority. Another real-life scenario where our system fits perfectly is smart traffic and security. In the modern world, security cameras (CCTV) are dispersed throughout the cities to ensure and monitor the safety of all citizens. However, these CCTV are computationally limited rather than both energy and computationally limited as the sensors in the above example. Since they are computationally limited, they depend on external resources to do the computation on their behalf. The CCTV can opportunistically take advantage of passing by vehicles to apply some face recognition algorithms on the frames it captures. Thus, it will be able to recognize the faces of suspected or wanted criminals. A car movie by down some street the CCTV situated on that street will send video frames to it to get analyzed, after some time another car recognizes the same face for the third party managing the CCTV are able to track the trajectory of this person throughout a populated area.

Chapter 3

Literature Review

In this chapter we survey the most relevant related work. We classify them to two main topics: vehicular offloading and edge-cell offloading.

3.1 Vehicular Offloading

Vehicular cloud is a distinctive kind of cloud server that is mobile and has high resource availability along with internet access where individual mobile devices can be either cloud users or service providers with a pay as you go model [27]. Vehicular clouds are a means of sharing resources between mobile entities where in some cases, this paradigm proves itself to be more efficient by keeping tasks locally instead of sending them to the cloud, which would be more expensive and require more time [28]. Different approaches have been discussed and proposed in the literature.

3.1.1 REPRO: Time-Constrained Data Retrieval for Edge Offloading in Vehicular Clouds [1]

A mobility oriented data retrieval protocol for computational offloading in vehicular edge computing was proposed in [1] to efficiently retrieve results of offloaded tasks by using vehicles and RSUs as relaying entities. In this paper, a hybrid of a topological-based protocol and a distance-based forwarding protocol were investigated. RSUs allocate tasks to suitable vehicles in the vicinity. After the vehicle does the computation, it checks if it is in the neighborhood of the RSU, if so it relays the result back to it. If the vehicle is not in the same area as the destination RSU, it should check if it in the range of any RSU. If that is the case, then the vehicle sends the result to the closest RSU. In its turn, the closest RSU will forward it to the destination RSU following the topology forwarding protocol. The RSUs are in constant communication with each other and their network topology is known so it is easy for a message to reach its destination once it reaches a collaborator in the RSU network (any RSU).

Although this looks like a promising approach, assigning the RSU the role to allocate tasks to nearby vehicles increases the probability of overwhelming the RSU with requests and increasing the burden on the fixed infrastructure, It is known that in urban areas the number of vehicles is huge which would be directly proportional to the number of requests that would be made. Moreover, no scheduling mechanism was implemented on the vehicles' ends which affects the availability of resources negatively. Additionally, the use of distance-based forwarding increases the probability of the loss of data since it depends on the opportunistic availability of vehicles reaching the RSU network. Thus, in this approach you'd be gambling on the randomness found in vehicles trajectories.

3.1.2 Mobile Edge Computing for the Internet of Vehicles [2]

A V2V offloading approach was proposed in [2], however, unlike the majority of the related works, this approach takes into account mobility as a criterion in order to select the processor vehicle. It motivates the use of a fixed infrastructure (RSU) to service computation requests. This is considered a setback as the number of vehicles might increase tremendously in a certain area, thus overwhelming the infrastructure.

3.1.3 Finding a STAR in a Vehicular Cloud [3]

A system where vehicles offer services to other cars (consumers) such as internet access was proposed in [3]. Ordinarily, RSUs handle such services. However, high demand in urban areas might lead to congestion of the network and lousy quality of service. Thus, in the proposed system, RSUs store, for each offering vehicle, the type of resources, their attributes, and the required price per resource unit. Beyond that, RSUs share the information they have with each other forming a dynamic registry together. Consequently, consumer cars request from the closest RSU the required resources, which, in turn, would find the best vehicle that can offer its services and satisfy the consumer's requests. Moreover, they also provide the option for consumer cars to select the offering vehicle they prefer according to their criteria. This dynamic registry can play the role of a proxy between the client (asking for services) and the provider (offering the services) in order for the client to find a resource it should contact the RSU network.

3.1.4 Computation Offloading Management for Vehicular Ad Hoc Cloud [4]

Similar to [3], an approach in which a car can offload some of its tasks to other selected vehicles, which offer their resources, was proposed in [4]. Four different selection strategies are put into the study to choose the best car for such computation. Multi-attributed selection strategy depends on various parameters to select a node such as computation capacity, longest communication time, communication overheads for transmissions per task, and the computing overheads for execution per job. This strategy outperformed all others that were analyzed with it, such as random selection, computation capacity-based selection, and distance-based selection strategy. To discover resources, a query packet is sent to all vehicles utilizing flooding. While the offloaded tasks are running on the designated vehicle, the approach continuously monitors the runtime status of the duties and reports to the client vehicle.

In order to increase the possibility of finding a resource, the proposed approach assumed that a query packet is flooded over the network where every vehicle that receives a packet will broadcast it. This assumption leads to the deterioration of the network service, where flooding the network would lead to congestion and drops especially in dense areas. Moreover, in the selection strategy, this approach doesn't take into consideration the quality of the connection between two entities.

3.1.5 VANET-Cloud: A Generic Cloud Computing Model For Vehicular Ad-Hoc Networks [5]

A new vehicular cloud model in which the vehicular clouds provide not only vehicular drivers but also other users with computing resources was proposed in [5]. It consists of three layers; client, communication, and cloud layer. The client of such a service can be a general customer, not necessarily a VANET entity or node. Using communication and computing devices such as smartphones, laptops, onboard computers, and GPS, the end-user can establish a service request to the adjacent layers. The communication layer consists of communication devices and networks such as internet gateways, wireless networks (VANETs), RSUs, and satellite GPS. At this stage, the client in the lower client layer and the communication devices used define the connection technology. This approach encompasses traditional cloud computing. An opportunistic model was proposed, however, the use of traditional cloud computing paradigm was not completely substituted since the opportunistic model's best fit is in rural areas where the connectivity is inherently limited.

3.1.6 Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing [6]

A task allocation mechanism across stationary and mobile fog nodes was formulated in [6] as an optimization problem with constraints such as service latency, quality loss, and fog capacity. Real-world taxi traces, was simulated to evaluate the effectiveness of the approach. The tasks implemented were video streaming and real-time object recognition. When compared with traditional fog selection strategies, the approach showed improved performance with respect to latency and quality balanced task allocation. In this approach, the client vehicle sends out a probe over DSRC to collect responses from available fog nodes in the vicinity. After discovering the fog candidates, the client vehicle sends a request to the zone head over LTE, which is a stationary RSU. When the zone head receives a request, it executes the task allocation algorithm to decide where to run the tasks. However, due to the mobility of the client and fog node, the connection between them may not last until the task is complete. Thus, the zone head of the current service zone must find another fog node to hand over the tasks to.

Briefly this approach depends on a zone head to coordinate between clients and providers which is considered a limitation in the proposed approach. Moreover, the incorporation of a zone head over LTE incurs an extra burden on the LTE network since, as 5G network are deployed, the number of devices is going to increase tremendously and more devices depending on the LTE network will increase, thus leading to overwhelming the network.

3.2 Edge-cells Offloading

Edge-cell offloading is a variant of opportunistic fog based computing where vehicles might offload their data to store them. In addition, other computationally limited devices might collaborate to service each other, or offload their data to fog nodes whether they are mobile or stationary.

3.2.1 Storage on Wheels: Offloading Popular Contents Through a Vehicular Cloud [7]

An approach that considers vehicles as cache edge data cells and controlled directly by the ISP was proposed in [7]. In this approach, devices might query directly (WiFi or 802.11p) other devices for faster retrieval of information instead of querying the infrastructure since the main aim in their approach is to minimize the load on the cellular infrastructure. Moreover, mobility was exploited in a sense where when a user requests content, if it is not immediately available in a nearby vehicle, the user agrees to wait for few minutes until any car with the content moves within range. The centralization of the decision in an ISP (such as RSU or stationary fog managed by the ISP) leads to negative effects on the network since too many requests might overwhelm it.

3.2.2 An opportunistic resource management model to overcome resource-constraint in the Internet of Things [8]

A decision-making model based on the object's characteristics such as computational power, storage, memory, energy, bandwidth, packet delivery, hop count was proposed in [8]. Depending on the characteristic, an IoT can choose the best nodes stationary cars that can be used as fog nodes and the best one is selected according to the characteristics. In this approach, stationary vehicles were utilized to act as fog nodes, however, this is not possible since stationary cars energy might get depleted while serving other devices. Moreover, the selection characteristics doesn't incorporate the signal strength as a criteria, because without it, drops might occur frequently since all the characteristics mentioned are not related to the quality of the connection but rather conditions that are related to the resources the vehicle is able to provide.

3.2.3 Opportunistic Fog for IoT: Challenges and Opportunities [9]

In order to support IoT application in hazardous environments, an opportunistic fog model that takes advantage of stationary and mobile fog nodes was proposed in [9]. In this approach, the concept of context-aware fog cluster was introduced. It is managed by a fog manager which is a fog node that has access to the cloud. Thus, the manager is responsible for collecting context data from the cluster members and performing analytics to infer cluster health, predict mobility patterns, detect rogue nodes, and broker quality of service agreement and pricing. Context data will include network information, environment data, rates of new nodes joining and existing nodes disconnecting, and node meta-data such as data about the node owners, node resources, task completion rates, and task accuracy. The cluster manager node's inferences and predictions are used to formulate policies related to task scheduling, node mobility, scaling, fault tolerance, security, and pricing with the cluster. The cluster manager will choose the appropriate cluster member to be used and if a better candidate exists, the role is migrated to it. The cluster manager learns from historical data and creates artificial intelligence (AI) models to study fog and edge behavior, emphasizing on calculating the probability of nodes to get disconnected and estimating node mobility.

The use of cluster management or a coordinator for resources as mentioned before incurs a trade-Off between the amount of network traffic and the possibility of overwhelming a single point of failure. Moreover, in this approach the role of a cluster manager can be assigned to another entity if it has better conditions, however, any disruption to the migration process might lead to the loss of all the data.

3.2.4 Data and Task Offloading in Collaborative Mobile Fog-Based Networks [10]

The problem of offloading and computation from a mobile device to the cloud to fog nodes or other mobile nodes in the vicinity was tackled in [10]. A layer composed exclusively of mobile devices that collaborate opportunistically, as a first resort for offloading, was added. Four different scenarios were studied: Device-to-Device (D2D), where mobile devices communicate with each other and spread their load to neighboring devices; cloud-only mobile devices that can only communicate with the cloud; D2D and cloud where mobile devices communicate to both neighboring mobile devices and cloud; fourth scenario, adds to the previous scenario fog nodes, which are located at the edge of the network. The four different scenarios were implemented and it was evident that the existence of a crowd computing layer below the fog nodes layer (drop computing) is beneficial and suitable for restricted mobile networks.

In this approach the devices depend on drop computing where the devices collaborate with each other before trying to offload. This is considered beneficial, however, the dependence on other computationally limited devices restricts the type of tasks that can be handled on the lower level before offloading to stationary infrastructure (cloud and fog) which are considered costly.

As the amount of data generated escalated and proliferated, the concept of offloading became an essential paradigm and a lot of research was put into it. It started with cloud computing, where data is offloaded to remote servers found in the cloud. However, it was coupled with latency, which affects time-centric tasks, which depends on receiving data in a timely manner. Due to the problem faced by cloud computing, the fog computing concept emerged to tackle this issue and decrease the congestion and latency at the cloud. The fog nodes bring cloud services to the edge of the network. They are machines with medium computer capabilities that can perform most of the tasks provided by the cloud. Although fog nodes are smaller than the cloud, they are not cheap. Here comes the concept of opportunistic vehicular computing, which is a cheaper alternative for fog computing and makes use of idle resources that already exist, which is cost-free and does not need additional infrastructure investment. The opportunistic vehicular fog computing is an infrastructure-less approach which is an excellent alternative for the fog computing paradigm since it also brings the services to the edge of the network and at the same time decreases the cost for service providers to provide services to consumers.

Chapter 4

Proposed Framework

In this chapter we give a detailed description of our framework, in addition, describe our architecture, and discuss the previously mentioned related literature and contrast them to our approach.

4.1 Basic Idea

The main limitation of related work is the centralization of decision and management of resources to one entity which is usually a fixed infrastructure. Moreover, some of the related work propose the use of WiFi which incurs an increased burden on the cellular network. In our work, we mitigate this issue by adopting DSRC as a means of communication. Also some related work flooded a network with requests such that every vehicle receiving a broadcast will broadcast the same message again until the whole network receives this message. This is unefficient in terms of network traffic and storage since every car will have to store information in its cache until a certain time. In our approach, we are proposing an edge cell opportunistic offloading framework. Indeed, the IoT devices are able to directly communicate with vehicles and utilize their idle resources. Hence the proposed framework covers the auto-arrangement of IoT devices as clusters, which was not tackled in the surveyed related work and the selection strategy depends on the SINR and the time needed for offloading data and retrieving the results. Also, we decentralize the decision making instead of assigning a device the role of a coordinator.

Briefly, instead of sending data from IoT devices to the fog for computation, in this thesis, we propose using vehicles as Mobile Computing Nodes (MCN) offering computational services as they roam around the cities. This approach can mainly be used in public vehicles such as taxis, buses, and delivery cars, which can offer these services to computationally limited bandwidth IoT devices and represents a profitable model.

IoT device owners would find it as a cost and delay efficient technique to offload computation instead of sending their data to the cloud or fog, which adopts the pay as you go model of payment and usually suffers from unreliable latency as stated earlier. Our proposed approach is adaptive andcan be applied in different domains. For example, in a planned or unplanned evacuation, there is possible damage to the mobile communication infrastructure. Thus, vehicular clouds might offer a temporary replacement for the infrastructure [27], which motivates the transition from regular clouds to vehicular clouds. Our proposed approach is adaptive and can be applied in different domains. 5G networks needs data offloading to offer higher bandwidths. Our proposed system would contribute to enhancing the user experience [41] [42].

4.2 Proposed Framework Architecture

In our approach the cars usually communicate with each other on the control channel over the DSRC to exchange safety messages and advertisements, the vehicle switches between control and service channels, whenever the car is in the control channel and willing to share resources it will broadcast a beacon message which is received by IoT devices stationed around it. Thus, advertising its availability, the IoT device in need for some computation to be done would send a reservation request after it calculates the SINR and the time of connection.

After that, the car sends an acknowledgment that it is successfully paired with this IoT device. Note that an IoT environment consists of many nodes thus we will cluster the IoT environments into small groups and each group is assigned a coordinator. This coordinator role is rotated among the participants in this cluster in a round robin fashion so that the burden of establishing a connection and sending data is shared among all the participants. Moreover, one of the problems that we would be facing is the choice of the best MCN by an IoT device that wishes to offload specific communication tasks to it. We plan to choose the top N MCNs within the range of the fixed IoT device, according to the SINR and the time of connection, taking into consideration the mobility of the MCN. Another challenge we should mention would be delivering the results back to the device or another party, knowing that the vehicles will not be stationary, which is explained below in the architecture.

4.2.1 Communication Protocol

In our approach, the network is composed of a numerous number of stationary IoT nodes that are dispersed throughout the city. They can be found in buildings (fire sensors) or standalone (cement health sensors). Moreover, in order to establish a network where we can send and receive data, these IoT devices will communicate with vehicles playing the role of MCNs roaming around the area. If these vehicles happen to be within the IoT device-Vehicle communication range, discovery, MCNs selection, and offloading (to a selected MCN) may occur.

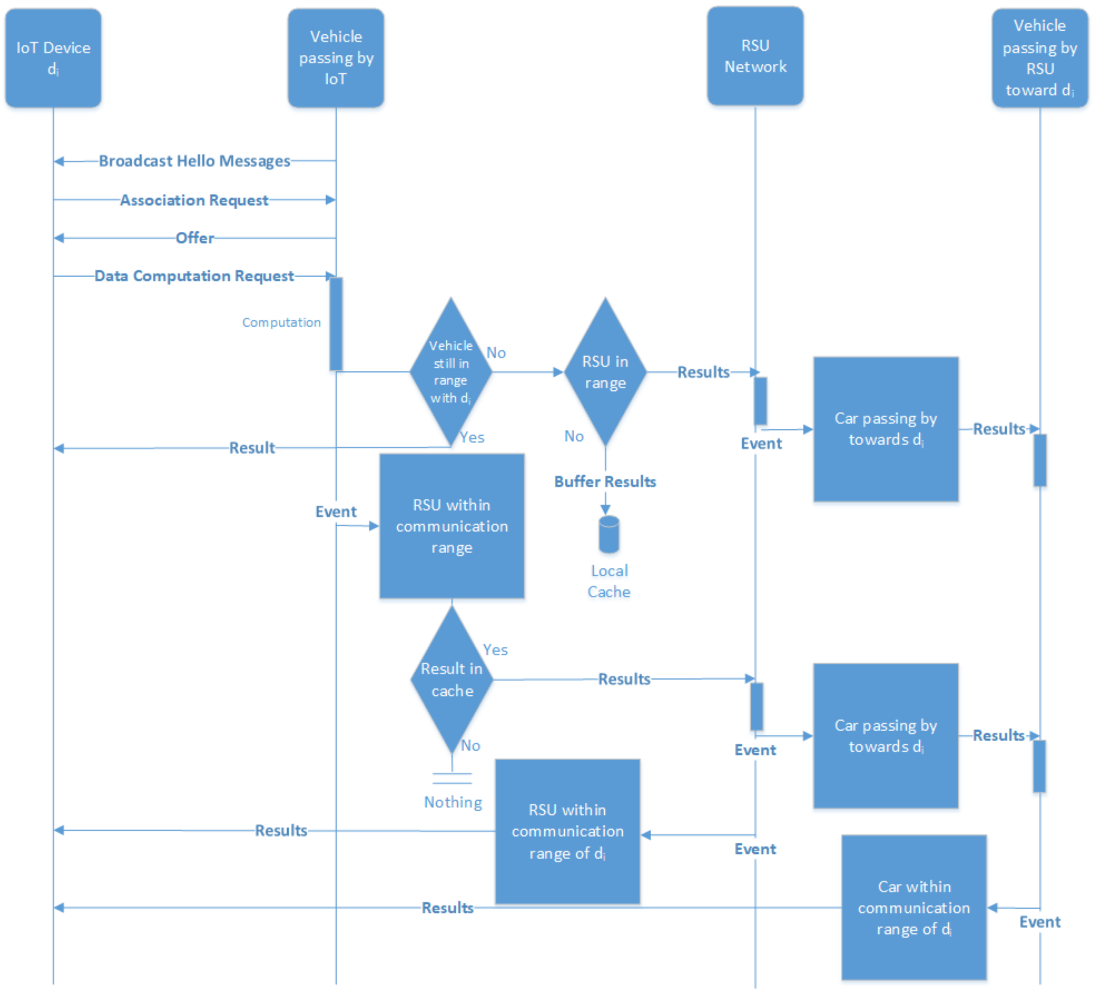


Figure 4.1: Communication protocol flow diagram

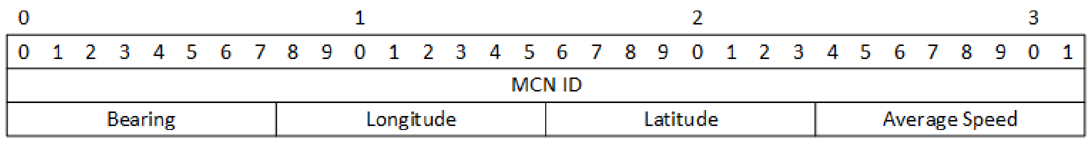


Figure 4.2: Hello Message packet

The communication protocol used in our approach is shown in Figure 4.1. In this protocol the cars usually communicate with each other on the control channel over the DSRC to exchange safety messages and advertisements. The vehicle switches between control and service channels. Whenever an MCN is in the control channel and willing to share resources it will broadcast a beacon/Hello message to IoT devices stationed around it. This hello message is illustrated in Figure 4.2 which consists of 4-byte field that represents the MCN ID (We assume MCN ID is an IP address); 1-byte field representing the bearing which is the direction of the car which can be calculated by using the longitude and latitude of the car; 1-byte field representing the longitude, another byte to represent the latitude; 1-byte field to represent the average speed of the vehicle. Thus, advertising the availability of vehicles to nearby devices.

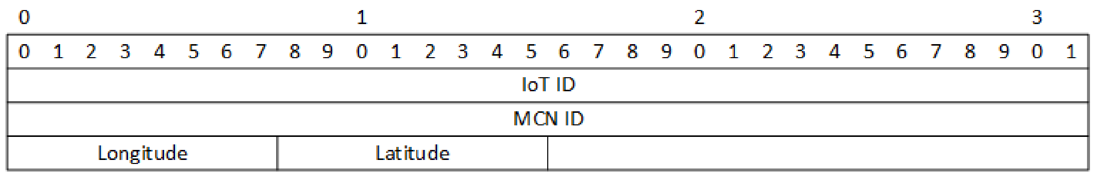


Figure 4.3: Association Request packet

The IoT device in need of some computation will reply to the Hello message by sending an association request after calculating the SINR and the time of connection and then selecting an MCN that has the best SINR and time of connection to be able to offload all the data. The association request contains a 4-byte field representing the IoT device ID (we assume it has an IP address); another 4-byte field to represent the MCN ID; 1-byte field for each of the latitude and the longitude as depicted in Figure 4.3.

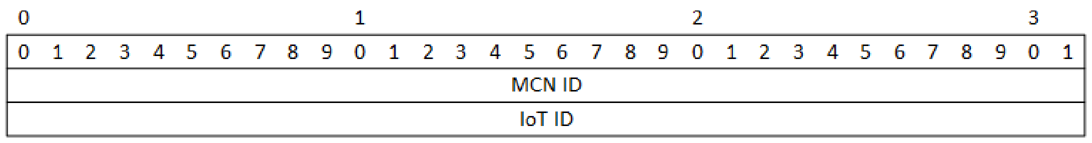


Figure 4.4: Offer Request packet

After that, the MCNs send an offer request that it is successfully paired with this specific IoT device this packet consists of the MCN ID and the IoT device ID shown in Figure 4.4. Note that an IoT environment consists of millions of nodes; thus, we will cluster the IoT environments into small groups, and each group is assigned a coordinator. This coordinator role is rotated among the participants in this cluster in a round-robin fashion so that the burden of establishing a connection and sending data is shared among all the participants. Creating clusters within IoT environments is essential for urban environments since it decreases the amount of data sent which leads to a decrease in the network congestion; moreover, it saves energy on the long run as proved by many previous researchers [43][44][45][46][47]. Furthermore, to select an MCN by an IoT device, we will choose the top N MCNs within the range of the fixed IoT device, according to the SINR and the time of connection, taking into consideration the mobility of the MCN if there exists more than one MCN offering their resources.

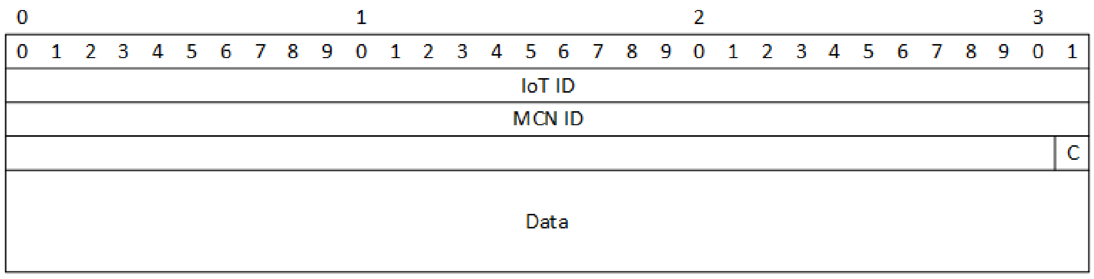


Figure 4.5: Data Computation Request packet

Upon choosing an MCN, the IoT device cluster head sends the data to it. This data computation packet, illustrated in Figure 4.5, contains a 4-byte field representing the IoT device ID and another 4-byte field representing the MCN ID; 1-bit field C is used to show the criticality of the data computation; and finally, the data that needs processing. If this 1-bit field is set, then the IoT device requires the data in a timely manner and cannot tolerate data loss. To deliver the results back to the device directly or through another party, four different scenarios might occur as illustrated in Figure 4.6. When the MCN finishes computation, it checks if it is still in the communication zone of the IoT device cluster head, if so it will send back the result to the IoT device. However, if the MCN has left the communication zone and the MCN can establish a connection to the RSU network, then the RSU network will relay the result back to the IoT device either directly if one of the RSUs is within range or through sending it to another MCN moving towards the IoT device, once this MCN is within range of the device it will offload the result back to the IoT device. Bear in mind the RSU network is multiple RSUs connected through wires, they are preconfigured by the service provider and each one knows its location

If the criticality field is set in the data computation request packet. Then the broker RSU will send the data to more than one MCN going in the direction of the IoT device for a specific duration, which we call dwell time. After the dwell time expires, the RSU network continues sending the data to MCNs going in the direction of the IoT device in a lesser rate until the IoT device sends back an acknowledgment to any MCN passing by which will deliver it back to the RSU network, and thus the broker RSU will stop on sending the data to near by vehicles. Also, if the IoT device doesnt receive the result within a certain duration, which is referred to as an expiration timer the IoT device will try using traditional techniques like cloud and fog computing or send the data to another vehicle.

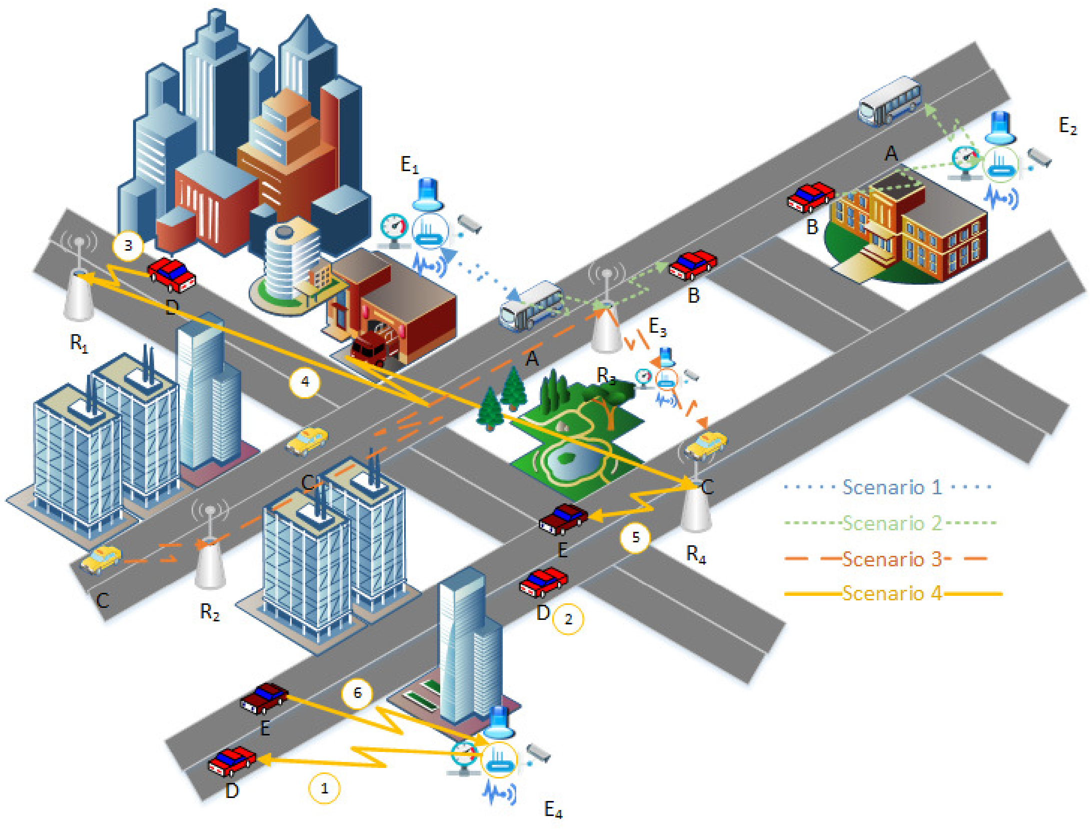


Figure 4.6: Overall Architecture displaying four different scenarios

After the MCN receives the data from the IoT device cluster head, four different scenarios might occur. Figure 4.6 shows an overview of network architecture. Shortly after receiving the data computation request from the IoT device cluster head and processing it, MCN$_k$ starts performing the task. When it finishes performing the task, MCN$_k$ checks if it is within the communication range of the IoT device. If is is, MCN$_k$ will directly send the result back to the cluster head which is depicted in Figure 4.6 as scenario 1. Scenario 1 shows vehicle A that is able to send the result immediately to the IoT device cluster head. Vehicle A will then continue with processing other task requests if there are any requests waiting in the buffer.

However, if MCN_k has left the communication zone, vehicle A will cache the results until it hears a "hello" message from the RSU_i. It will then relay the result to RSU_i. As soon as the RSU_i receives the result, it will check with the RSU network manager for the closest RSU in terms of driving distance. If the IoT device cluster head is not within communication range of any RSU, it will relay the result to the closest RSU, which will cache the result until an MCN passing by is moving toward the IoT device cluster head. Each RSU registers its location with the RSU network management system and maps the simulation area upon initialization. Through this process, the RSU manager obtains information about the road and RSU network structure. Scenario 2 shows vehicle B that receives a task from the IoT device cluster head of environment E_2 and then leaves the communication zone. It then finishes the computation after a certain time and hears a "hello" message from R_1. It thus offloads the result to the RSU network through R_1 which is its portal to the RSU network. After consulting with the RSU manager, RSU_1 finds that R_2 is closer to E_2 in terms of driving distance, so it forwards the result to R_2. R_2 now has two choices. If E_2 is not within communication range of R_2, R_2 will relay the result to a vehicle with predefined trajectory (vehicle C).

In scenario 3, if the IoT environment E_1 is within the communication range of an RSU (R_1), the RSU will directly forward the data to the IoT device cluster head. Finally, in scenario 4, if a vehicle offloads the result to an RSU (R_2) and this RSU finds itself as the closest, it will just cache the result until a vehicle passes by moving towards environment E_2. Scenario 4 works under the impression that the IoT environment E_2 is outside the communication range of the RSU R_2.

Scenario 4 consists of 6 steps depicted in Figure 4.6:

1. The MCN and the IoT device cluster head establish a connection. It broadcasts every 1 second (during the control channel) when the MCN is available, thus decreasing the amount of network traffic, which later affects the SINR. If the IoT device gets more than one broadcast, it will calculate the time of connection and the SINR for each one and replies to the most suitable MCN. Thus, the MCN replies with an offer request confirming the

pairing between the two entities. After that, the IoT device sends a data computation request which contains the data.

2. After the MCN receives the data, it starts performing the computation while it is roaming around the city.
3. The MCN offloads the result to the R1, R1 checks if it is within communication range with the IoT environment E4
4. If R1 is not in the communication range, it checks within the RSU network if there are any closer RSUs to the IoT device location. Where R1 found that R4 is closer to the IoT environment E4. Thus, it will relay the data to R4 which will be assigned as the broker responsible for delivering the data back to E4
5. Consequently, E4 checks if it is within the communication range of E4. If not, it will cache the data in local storage until it finds a car moving in the direction of E4 and relays the data to it.
6. Once the MCN is within the communication range of E4, it will offload the result back to the IoT device coordinator, see step 6, which will handle the result and take the proper action according to it.

4.2.2 Time of IoT device-MCN Connection

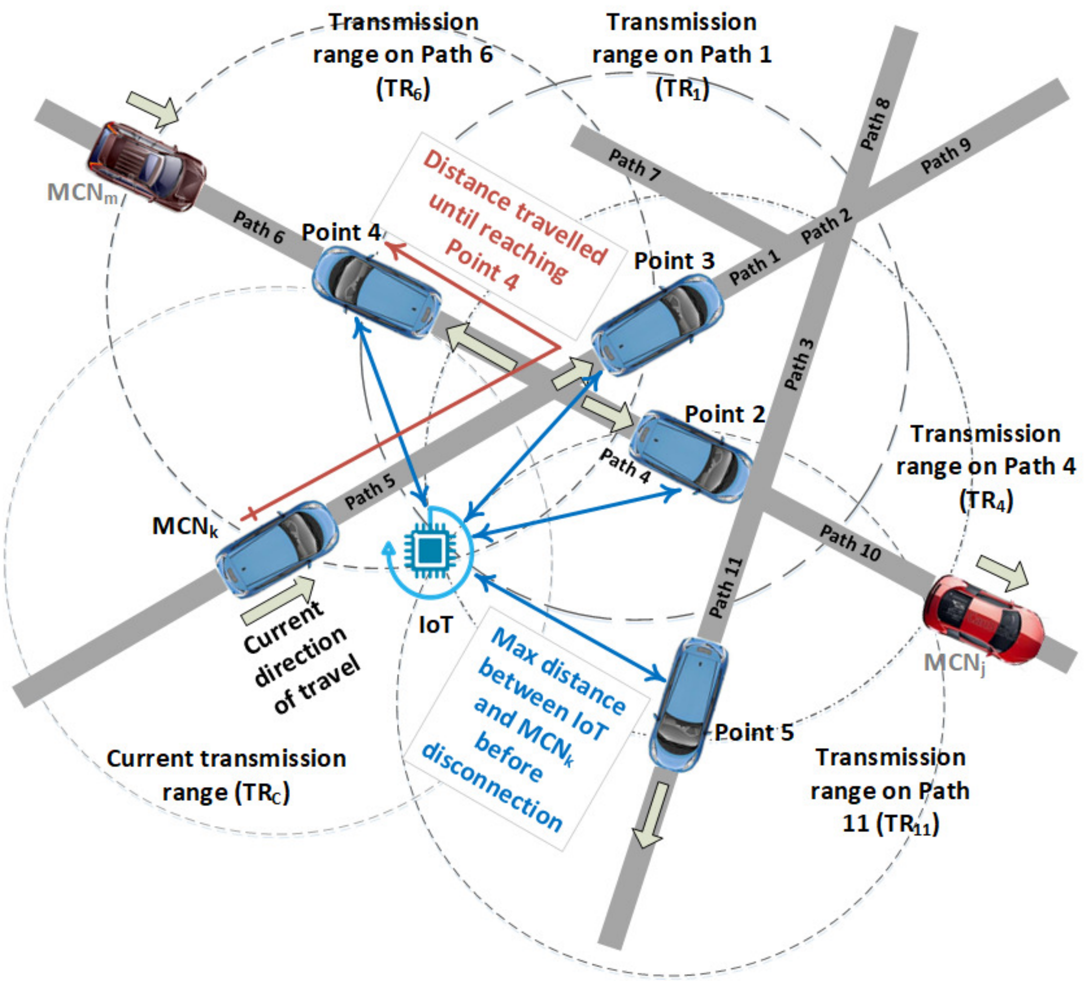

Transmission range on Path 6 (TR6)
Transmission range on Path 1 (TR1)
Path 8
Path 9
Path 7
Path 2
Path 1
MCNm
Path 6
Point 4
Distance travelled until reaching Point 4
Point 3
Path 3
Point 2
Path 4
Transmission range on Path 4 (TR4)
Path 5
MCNk
Path 10
IoT
Path 11
MCNj
Current direction of travel
Max distance between IoT and MCNk before disconnection
Point 5
Current transmission range (TRC)
Transmission range on Path 11 (TR11)

Figure 4.7: MCN paths of roads showing its farthest locations before it disconnects from the IoT device

We consider a macro-cell deployment where certain vehicles will play the role of mobile computing nodes. For the most part, these will be public vehicles, like taxis and buses, whose owners may be compensated based on the amount of computation they perform. A computing vehicle is equipped with a navigation system that associates the global positioning system (GPS) positions to road maps to enable them to know their locations (i.e., geometric coordinates). Similar to [48] [49], we calculate the time of connection (TOC), where the UE in [48] [49] resembles the lightweight IoT device we describe in our work. In our work the vehicle (not IoT device) calculates the time of connection unlike in [48] [49], to lessen the computation burden on the IoT device taking into consideration that the IoT device does not know its surroundings or environment. After the

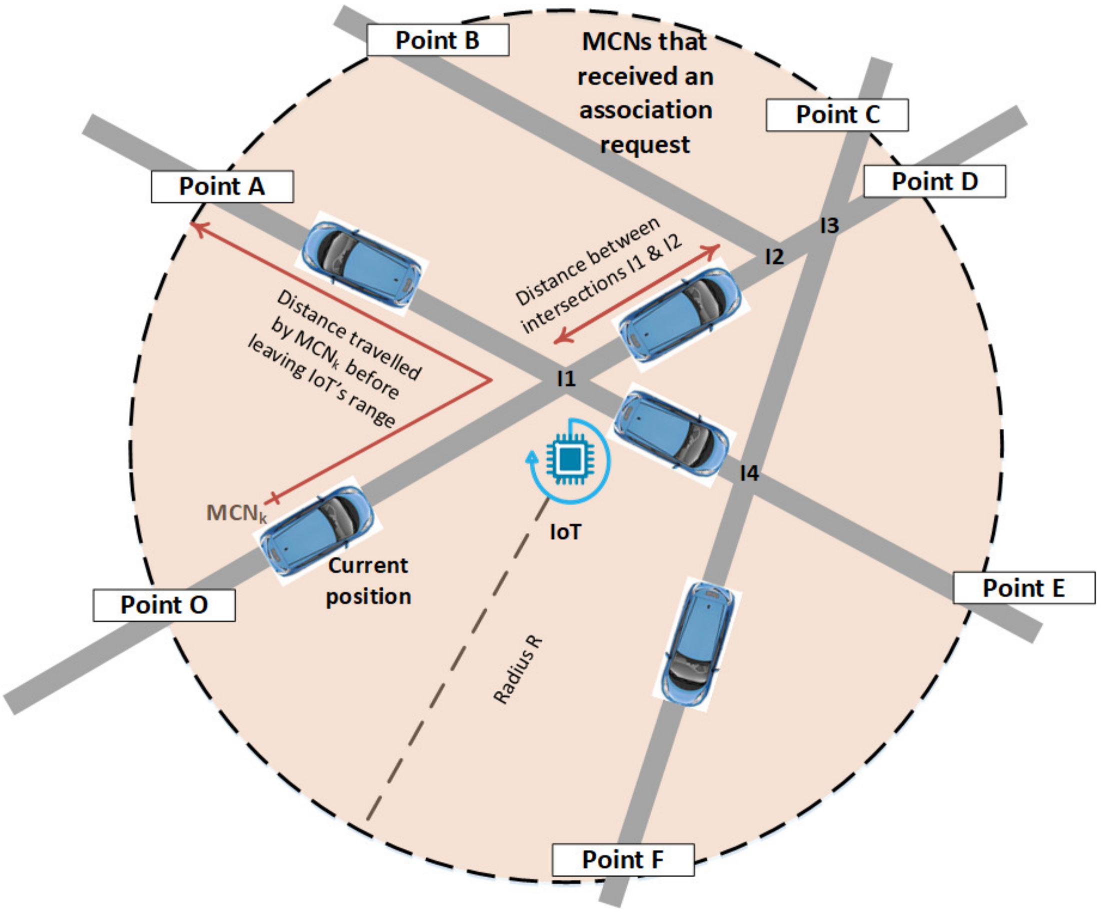
MCN receives the association request, which includes the IoT device position and other information, it will send an offer request. The offer request is sent only if the time of connection calculated by the MCN on behalf of the IoT device is sufficient for the IoT device and the vehicle to associate with each other (TOC $>$ time threshold).

A simple general scenario is illustrated in Figure 4.7, where each MCN vehicle periodically broadcasts the beacon "hello" message which includes the MCN vehicle's position. This information allows the IoT device to know if the MCN is within the communication range. If it is not, the IoT device will not reply because the IoT device packet will never reach the vehicle since the transmission range of an IoT device is much smaller compared to that of an MCN. Figure 4.7 shows the current location of an MCN (MCN_k) in the bottom-left region of the map, along with its possible farthest positions (Points 2, 3, 4, and 5) before the IoT device disconnects from it due to exiting its transmission coverage. The transmission coverage of MCN_k in the different positions is illustrated using dotted circles, while its range is depicted through the two-headed arrows that connect the IoT device.

In our work, we consider two metrics for the IoT device to pair with the most suitable MCN: Signal-to Interference and Noise Ratio (SINR) and the expected Time of Connection (TOC).

The MCN can change its path or direction after reaching an intersection. Depending on which road segment the vehicle is on, the time of connection (TOC) between an IoT device and an MCN might vary widely. The example in Figure 4.7 shows MCN_k, in range with an IoT device, is approaching the intersection at the center of the map. After reaching the junction, the MCN can take one of the three possible routes, each of which results in a different TOC. After crossing 2, 3, 4, and 5 on paths 4, 1, 5, 6, and 11, respectively, the MCN_k loses IoT device coverage and becomes out of the communication zone of the IoT device.

Given the MCN location along with the IoT device location, the MCN can compute the coordinates of the exit points 2, 3, 4, 5 after receiving IoT device position. It can then compute, using its knowledge of the road map (by the aid of the GPS), the driving distances to each of these points. Each MCN can calculate its average speed; thus, it can translate these distances into times. Out of these estimated times, the expected time of connection could be calculated and consequently the MCN will send to the IoT device an offer request if the TOC is sufficient. Thus, the IoT device is aware that the MCN it received an offer from, is suitable in terms of the time of connection.



Point B
MCNs that received an association request
Point C
Point D
Point A
I3
I2
Distance between intersections I1 & I2
Distance travelled by MCN$_k$ before leaving IoT's range
I1
I4
MCN$_k$
IoT
Current position
Point O
Point E
Radius R
Point F

Figure 4.8: Zoom-in of Figure 4.7

Moreover, Figure 4.8, which is a zoom-in of Figure 4.7, describes the concept of time of connection. In this figure, MCN$_k$ is currently within the IoT device's communication range and approaching intersection I$_1$, after which it will take one of the three possible paths. An MCN is equally likely to take any path upon reaching a junction. However, different probabilities can be assigned as a function of some measurements or statistics like average traffic density on each of the outgoing paths. Such data may be obtained from the traffic authority database, or learned by RSUs which periodically receive beacon messages from MCNs passing by.

Consider:

M: Number of exit points
P: Number of distinct paths from the IoT device to exit points

We assume the MCNs have the same transmission range R$_{MCN}$, and so when an MCN needs to calculate its estimated TOC, it draws a circle having a radius R$_{IoT}$ and the IoT device coordinates as a center. It identifies the intersection points of this circle with the roads in range. As shown in Figure 4.8, the points are A, B, C, D, E, and F. They represent exit points beyond which the MCN will lose connection with this IoT device, and can be referred to as the set $\{p_1, p_2, \ldots, p_M\}$. Next, we identify the set of possible distinct paths from the MCNs location to the exit points and their corresponding lengths in meters $\{l_1, l_2, \ldots, l_P\}$, where $P \geq M$.

$$t_i = \frac{l_i}{AverageCarSpeed} \tag{4.1}$$

Given the cars average speed (including stopping at red lights and stop signs and slowing down on turns), there will be corresponding P times, i.e., $\{t_1, t_2, \ldots, t_P\}$, which are obtained by dividing the lengths of the paths calculated by the average speed of the MCNs as depicted in equation 4.1. Next, given that the MCN will likely take any road upon reaching an intersection with equal probability, the average (expected) time that it will take until it disconnects from the IoT device is:

$$TOC = \frac{t_1 + t_2 + \ldots + t_P}{P} \tag{4.2}$$

We refer to this value that is specific to each MCN as the expected time of connection (TOC). The possible paths that the MCN can take starting at its current position can be modeled as a directed acyclic graph (DAG) traversal problem. In this DAG, the vertices are the MCN's current position (its position when the MCN has to send an offer request for the IoT device to make an association decision), the set of road intersections, and the set of exit points. For example, in Figure 4.8, the vertices are the current position of the MCN, the intersections are $I1, I2, I3$, and the exit points are A, B, C, D, E. In the figure, it can be shown that the number of possible paths that the MCN can take to all exit points is 9 (i.e., $P = 9$).

The MCN, in proximity with the IoT device, computes the expected total time of connection and sends an offer message to the respective IoT device only if the TOC exceeds a certain threshold. It is important to note that if the IoT device receives an offer message, this indicates that the time of connection is sufficient. Therefore, the decision of the IoT device in choosing an MCN is defined by a minimum SINR threshold and the connection time to minimize the probability of disconnections and to increase throughput.

4.2.3 IoT device-MCN association

We begin this section by describing the communication protocol between the MCN and the IoT device. This protocol is summarized in Figure 4.9 and with the following steps:

1. It begins by the MCNs periodically transmitting connection or service (CS) advertisements in which they include information about their position and availability (e.g., percent of the queue that is occupied).

2. After receiving the MCNs advertisement, an IoT device that wishes to compute some data checks the SINR level, if the SINR level is higher than the threshold.

3. It sends to the MCN an association request that includes information about the data to compute along with its location

4. the MCN accepts the request if it has enough resources to handle the association request (e.g., if it deems that the size of the data is acceptable).

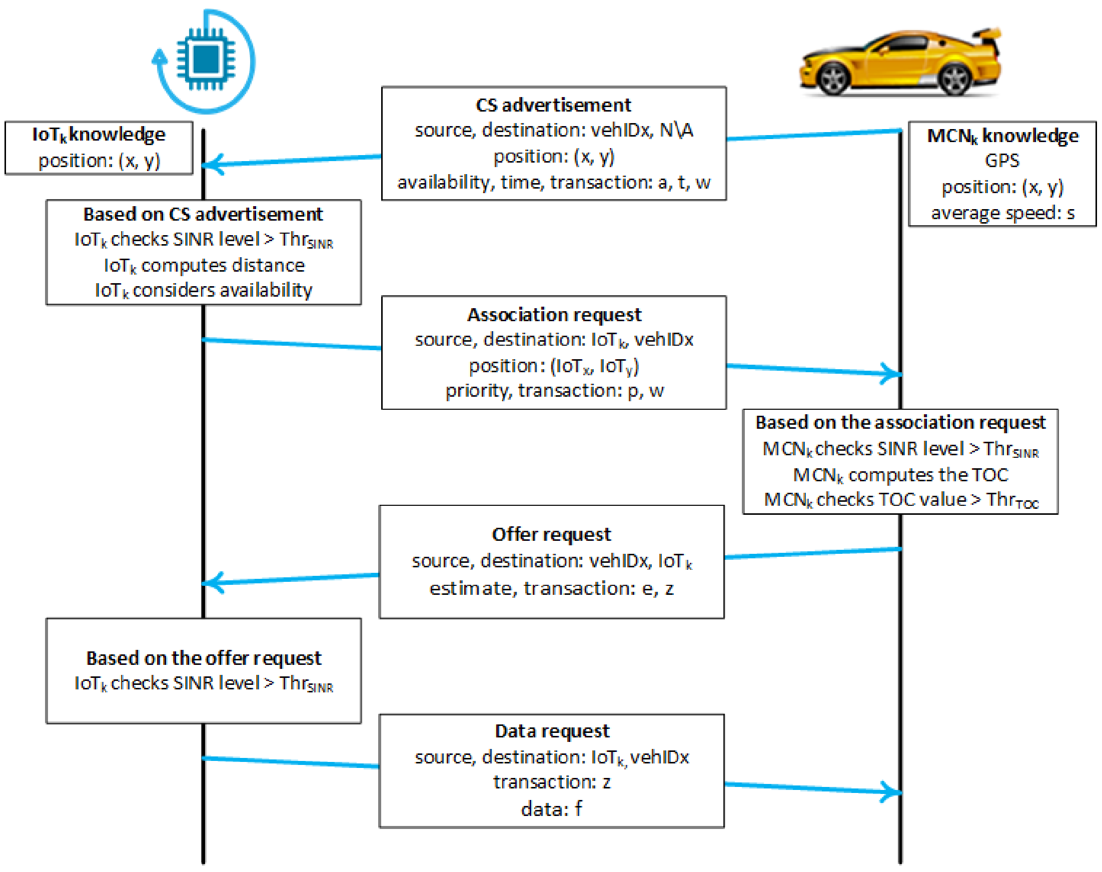
5. Consequently, if the SINR upon receiving the association message by MCN$_k$ is above a certain threshold, then MCN$_k$, using its knowledge of the environment through the GPS, along with its position and its average speed can calculate the TOC and checks if it exceeds a predefined threshold.

6. In its offer packet, the MCN includes an estimate of the time that the request waiting time before it gets serviced.

7. Then the IoT device, upon receiving the offer packet it will check the SINR level and implicitly knows that the TOC is sufficient to offload the data to MCN$_k$.

8. Finally, the IoT device sends the MCN the data to do some computation, thus implicitly accepting the MCNs processing delay.

Unlike [48, 49], in which the TOC is calculated by the IoT/user device, in our framework this task is performed by the MCN since it posses all the needed environment and mobility knowledge, along with IoT device position which piggy backed to the association message sent to the MCN as shown in Figure 4.9.



IoT$_k$ knowledge
position: (x, y)
CS advertisement
source, destination: vehIDx, N\A
position: (x, y)
availability, time, transaction: a, t, w
MCN$_k$ knowledge
GPS
position: (x, y)
average speed: s
Based on CS advertisement
IoT$_k$ checks SINR level > Thr$_{SINR}$
IoT$_k$ computes distance
IoT$_k$ considers availability
Association request
source, destination: IoT$_k$, vehIDx
position: (IoT$_x$, IoT$_y$)
priority, transaction: p, w
Based on the association request
MCN$_k$ checks SINR level > Thr$_{SINR}$
MCN$_k$ computes the TOC
MCN$_k$ checks TOC value > Thr$_{TOC}$
Offer request
source, destination: vehIDx, IoT$_k$
estimate, transaction: e, z
Based on the offer request
IoT$_k$ checks SINR level > Thr$_{SINR}$
Data request
source, destination: IoT$_k$, vehIDx
transaction: z
data: f

Figure 4.9: Communication protocol between MCN and IoT device

The task of computing the time of connection has been assigned to the MCN for another important reason which is our initial concern in this thesis. The IoT device, as we consider it, is a computationally limited and energy limited resource. Thus, to alleviate the burden of computing the time of connection from the IoT device, the MCN has been tasked with this. We must re-emphasize that the IoT device, to begin with, does not have all the information, in relation to the requirements of computing the TOC, needed to be able to proceed with such computations.

4.2.4 RSU Forwarding Mechanism

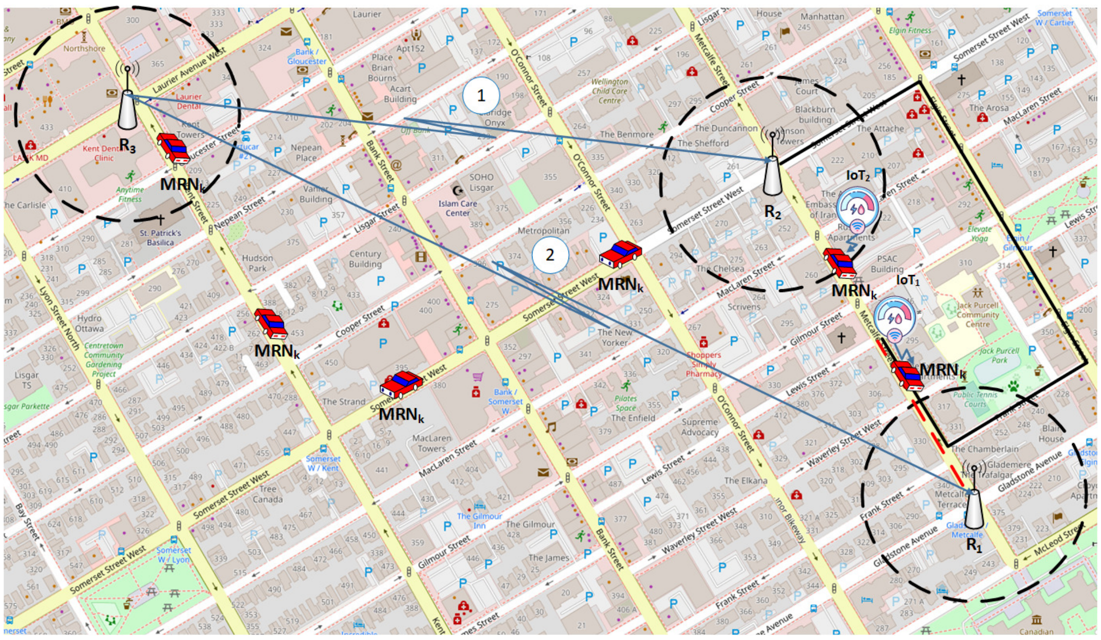


Figure 4.10: RSU forwarding

The RSUs have information about the running road traffic. Hence RSUs can map the region as a graph and find the shortest path using Dijkstra's algorithm. As illustrated in Figure 4.10, after MCN$_k$ has left the communication zone of the IoT devices and reached a distant RSU (R$_3$). R$_3$ will consult the RSU network to check if the IoT device lies in any of the RSUs' communication zones (RSU-IoT device communication zone). In scenario 1 in Figure 4.10, where IoT$_2$ is within communication range of R$_2$, R$_3$ will forward the result to R$_2$ which in its turn relays the information to IoT$_2$. Otherwise, R$_3$ will consult the RSU network to find the closest RSU in terms of driving distance. The RSU finds the shortest driving distance to the IoT device cluster head using Dijkstra's algorithm.

In scenario 2 in Figure 4.10, where you can see that for IoT$_1$, R$_1$ is closer in terms of driving distance where the dashed path from R$_1$ is much shorter than the route from R$_2$ (black route). Thus, R$_3$ will forward the result to R$_1$. Once R$_1$ receives the results, it will cache the results and check if it can directly relay the result to the IoT device. However, if it cannot directly contact the IoT device cluster head, and to assure that the information will not get lost, the RSU upon receiving a result will wait until a vehicle with a predefined trajectory/route (public cars) heading towards the IoT cluster head passes by. The suggested mechanism will ensure that the information will not get lost. This decreases unnecessary network traffic and congestion which positively affects the network's performance.

4.3 Discussion

Although in our approach, we adopted the vehicular cloud concept, where we offload tasks from stationary IoT devices to mobile vehicles. We considered cars as edge-computing cells, unlike [7], nearby IoT devices might contact an MCN directly using 802.11p for a more reliable connection. Also unlike [4, 3, 50], we took into consideration the SINR when selecting an appropriate car to perform a task or computation, and the time of connection, bearing in mind the mobility of the MCN. Furthermore, IoT devices, our consumers here, can directly discover MCNs and choose the best one instead of crowding the RSU's registry, which would lead to a delay in the network. In addition, we did not consider the computation power because we assume that all vehicles have the same computation power and computation resources to offer. Beyond that, we tackled some of the challenges that VANET clouds suffer from, which are not dealt with in [5], which are resource allocation and sharing, communication, and coordination between VANETs and clients.

Furthermore, the high mobility, SINR, unstable communication links, and efficient selection mechanism of participants, which in our case depend on two main factors, the time of connection and SINR, are still considered as challenges [27] [29] . Unlike [2, 7] Mobility in our approach depends on more than the mere speed of a vehicle but also on the direction and the time of connection.

Table 4.1 shows the features of our approach compared t other related works. [1] tackled offloading to vehicles to decrease network and RSU congestion. However, it considered vehicles RSUs, plus mobile phone users who want to do the offloading. We, on the other hand, have an environment of IoT devices (clusters of large numbers of devices that are characterized by small packets and periodic or event-driven transmissions) and look for MCNs that are public vehicles. Along with that, we exploited the fact that nearby IoT devices may have similar traffic and transmission patterns for group communication. Also, the mobility model in [1] does not allow to calculate the expected time of connection in which finding the best vehicle that will stay in the range of the IoT device for the maximum time. Plus, it did not considering the quality of the received signal (SINR) at each IoT device, but we will.

Moreover, [6] suggested offloading from vehicle to other vehicles and using the RSU as a coordinator and cluster manager without taking into consideration the SINR and time of connection. Also, if a vehicle did not finish its computation, it hands over the computation to the infrastructure, which increases the network traffic at the infrastructure. In [9] and [10], consider mobile devices and IoT devices offloading to MANETs that are energy limited. However, only [9] takes into consideration a coordinator who orchestrates the offloading mechanism, which is a fog or a mobile node, and perform some form of context-aware clustering. Although they consider the quality of service when choosing a node to offload to, however, they disregard the time of connection. On top of all of that, none of the discussed related work has clustered IoT devices and chose a cluster manager.

	Mode	Clustering	TOC	SINR	NTH	ITH	TRH	RA
Our Approach	I\F2V	✓	✓	✓	✗	✗	✓	✓
[1]	V2V	✗	✗	✗	✓	✗	✓	✓
[2]	V2V	✗	✓	✗	✗	✗	✗	✗
[3]	V2V	✗	✗	✗	✗	✗	✓	✓
[4]	V2V	✗	✗	✗	✗	✗	✓	✗
[5]	V2V	✗	✗	✗	✗	✗	✗	✓
[6]	V2V	✓	✗	✗	✓	✓	✗	✗
[7]	I2V	✗	✗	✗	✗	✗	✗	✗
[8]	I2V	✓	✗	✗	✗	✗	✗	✗
[9]	I2M	✓	✗	✓	✗	✗	✗	✗
[10]	M2M	✗	✗	✗	✓	✓	✗	✗

Table 4.1: Literature summary

The table shows the concepts we have implemented in this work. The occurrence or lack thereof of these concepts shows the difference between the related work and our work. The second column contains the mode of communication in relation to the entities communicating, the third column contains the use or lack of use of the clustering mechanism in the general architecture, the fourth and fifth columns contain the use or lack of use of the time of connection (TOC) and signal interference noise ratio (SINR), the sixth and seventh columns contain the use or lack of use of the node task handover (NTH) and infrastructure task handover (ITH), the eighth column contains the use or lack of use of the task result handover (TRH), finally the ninth column contains the use or lack of use of the the RSU assistance (RA).

Chapter 5

Performance Evaluation and Results

In this chapter, we describe the setup of our environment, and the tools used to implement our communication protocol. Furthermore, we evaluate our proposed approach in terms of various metrics such as retrieval success rate and retrieval delay.

5.1 Environment Setup

Offloading has been recognized as an effective technique for enhancing the battery life of IoT devices. It is also considered as an effective technique for lessening the burden of computation on them. Indeed, without offloading assistance, computationally limited devices and energy limited devices may suffer from extreme energy loss. This will happen if these devices try to perform some light or heavy computations on their own depending on the capacity of the device.

Compared to traditional offloading paradigms, mobile computing nodes, on the other hand, represent a more efficient solution, given the lack of need for installation and figuring out the logistics for deployment scenarios. Moreover, mobile computing nodes add flexibility since they can serve mobile users who may be located anywhere from urban areas to rural neighborhoods. On top of that, mobile computing nodes will decrease the load on the LTE network. For serving IoT devices, in terms of computational services, careful analysis needs to be performed to choose computing nodes that offer the best IoT and MCN association.

5.1.1 Simulation Tools

To conduct our simulations, we set up our environment, as indicated in table 5.1 below. The programming language used to implement our communication protocol is C/C++ using the ns-3 simulator.

Operating system	Ubuntu 18.04.4 LTS
System type	64-bit operating system
Installed RAM	32 GB
GNU Compiler Collection (gcc)	version 7.5.0
Network simulator	ns-3: 3.30
Traffic simulator	SUMO 1.4.0
Network and traffic simulator coupler	Vodafone Chair ns-3 - sumo coupler

Table 5.1: Environment specifications

5.1.1.1 Network Simulator 3 (ns-3)

We used the ns-3 simulator [51] to simulate the scenarios we mentioned above in chapter 4 (section 4.1). ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. It is a free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

5.1.1.2 Simulation of Urban MObility (SUMO)

Along with ns-3, we used "Simulation of Urban MObility" (Eclipse SUMO) [52]. SUMO is an open-source, highly portable, microscopic, and continuous road traffic simulation package designed to handle large road networks. SUMO is licensed under the Eclipse Public License V2. "Eclipse SUMO" is a trademark of the Eclipse Foundation. However, for ns-3 and SUMO coupling, we used the TraCI module implemented by Vodafone Chair Mobile Communcations Systems [53], which is an ns-3 module that implements a bidirectional coupling to the road traffic simulator in SUMO. It dynamically synchronizes the positions of SUMO vehicles with corresponding ns-3 nodes.

Additionally, the state of SUMO vehicles can be controlled via ns-3, e.g., for changing the speed. The module is built on top of the TraCI API of the SUMO simulator. The module prerequisites a SUMO installation of version 1.1.0 or more.

5.1.2 System Description

The simulation area used for our evaluation of the retrieval rate, and rate of serviced IoTs in this thesis is adopted from [1], which is a map of the downtown area of Ottawa, Canada (6421 m^2). Twenty-one RSUs were distributed with a separation distance of approximately 300m, as exemplified in Figure 5.1. All our service cars that are inserted into this simulation are public vehicles that have predefined trajectories and schedules from buses following a bus schedule to taxis with predefined routes.

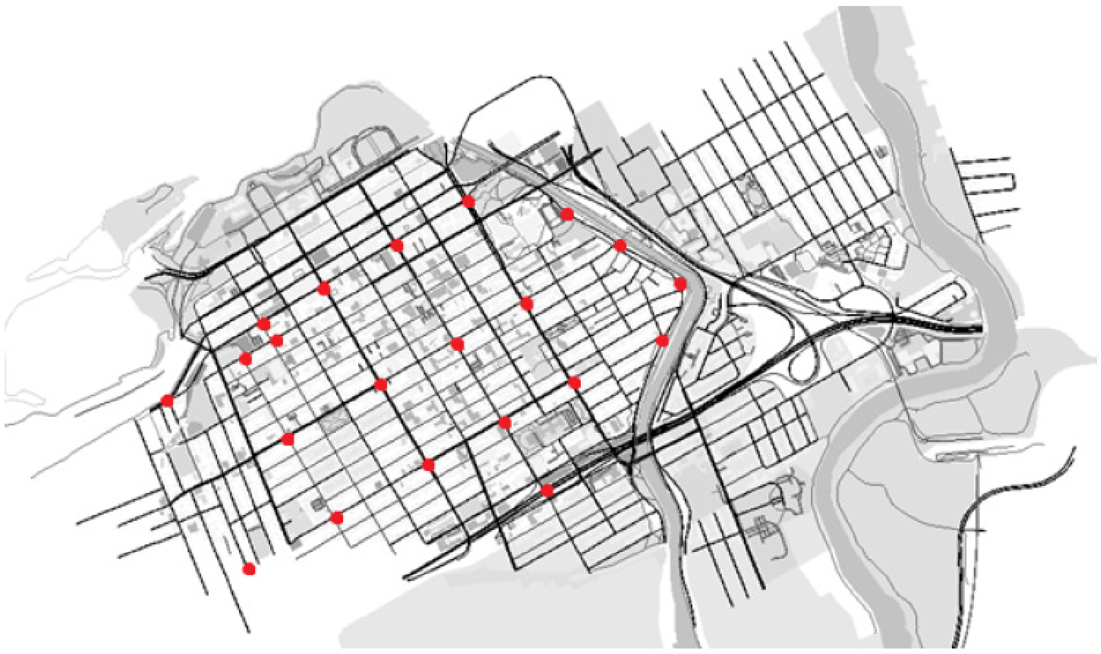


Figure 5.1: Ottawa Urban Center [1]

We list in Table 5.2 the used simulation parameters in our DSRC system and their default values, as inferred from [54] and [55] for most of them.

Parameter Name	Default Value
RSU power (dBm)	46.0206
MCN power (dBm)	46.0206
IoT power (dBm)	5
RSU urban transmission range (m)	145
MCN urban transmission range (m)	145
IoT urban transmission range (m)	40
RSU number	21
MCN number	20-100
IoT number	17-337
SINR threshold (dB)	7
Time of Connection threshold (s)	4
Simulation Time (s)	3600

Table 5.2: Simulation parameters with their default values

As suggested in [54], we aim to comply with the standardization of VANET simulation on ns-3. We ought to use the Log Distance propagation loss model since it was identified to be more appropriate for the urban environment. We ought to also use Two Ray Ground model in simulating rural environments with fewer obstructions. We also added fading, which is an essential means to make a simulation of propagation more realistic, so we used Nakagami-m fading. The parameters used for the propagation loss and fading are listed in the Table 5.3 and Table 5.4.

U/R	Propagation Loss Model	Parameter	Value	Units
Urban	Log Distance	ReferenceLoss	37.35	dB

Table 5.3: Summary of propagation loss model parameters

U/R	Fading Model	Parameter	Value	Units
Urban	Nakagami-m	m0	1.5	unitless [1]
Urban	Nakagami-m	Distance1	60	meters
Urban	Nakagami-m	m1	0.75	unitless [1]
Urban	Nakagami-m	Distance2	145	meters
Urban	Nakagami-m	m2	0	unitless

Table 5.4: Summary of fading model parameters

Furthermore, in our simulation, we study the communication between IoT cluster heads and the MCNs. It is assumed that the clustering mechanism in the background is fully functional, where IoTs use the Zigbee protocol to communicate with each other and share data.

The IEEE 802.15.4 specification is the basis of the Zigbee protocol [56]. It is a technical standard that defines the operation of low-rate wireless personal area networks (LR-WPANs).

The technology defined by the Zigbee specification is more straightforward and less expensive than other wireless personal area networks (WPANs), such as Bluetooth or more general wireless networking such as Wi-Fi. Due to its low power consumption, transmission distances are limited to 10-100 meters line-of-sight, depending on power output and environmental characteristics. Moreover, using a mesh network of intermediate devices, Zigbee devices can transmit data over long distances. Zigbee is used in low data rate applications that require long battery life and secure networking (128-bit symmetric encryption keys secure Zigbee networks).

The maximum MAC frame size defined by the IEEE 802.15.4 standard is 127 bytes, 25 used for frame overhead, and 102 for payload. The header for security purposes at the Link layer consumes up to 21 additional bytes, limiting the available space to 81 bytes for an IPv6 packet. Since the IPv6 header is 40 bytes long, it will result in 41 bytes for upper layers. Additionally, UDP is used which would leave only 33 bytes available for application data. The number of bytes remaining is not enough. Header compression based on HCI encoding allows 6LoWPAN to compress the 40 bytes of the IPv6 header using just 2 to 7 bytes. Thus header compression based on HCI encoding is adopted. Moreover, UDP provides a connection-less service with no guarantee at all. The respective

[1]Indicates a default value

header size is only 8 bytes. The impact on the packet size left free for upper layers protocols is shown in Figure 5.2 [11].

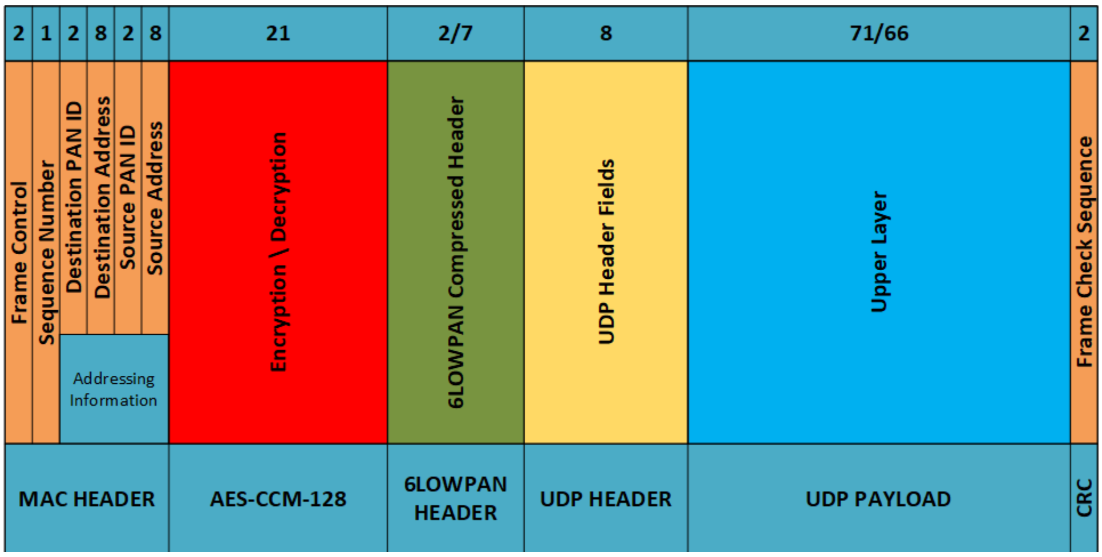


Figure 5.2: Zigbee packet format [11]

As indicated in [12], packets sent between DSRC devices have the format shown below in Figure 5.3. The MAC header consumes 14 bytes, along with 40 bytes occupied by IPv6 header plus 8 bytes added as a result of the UDP header. However, the Maximum Transmission Unit (MTU) in DSRC protocol is 1500 bytes, as suggested by [57], which results in 1438 bytes as payload available for upper layers.

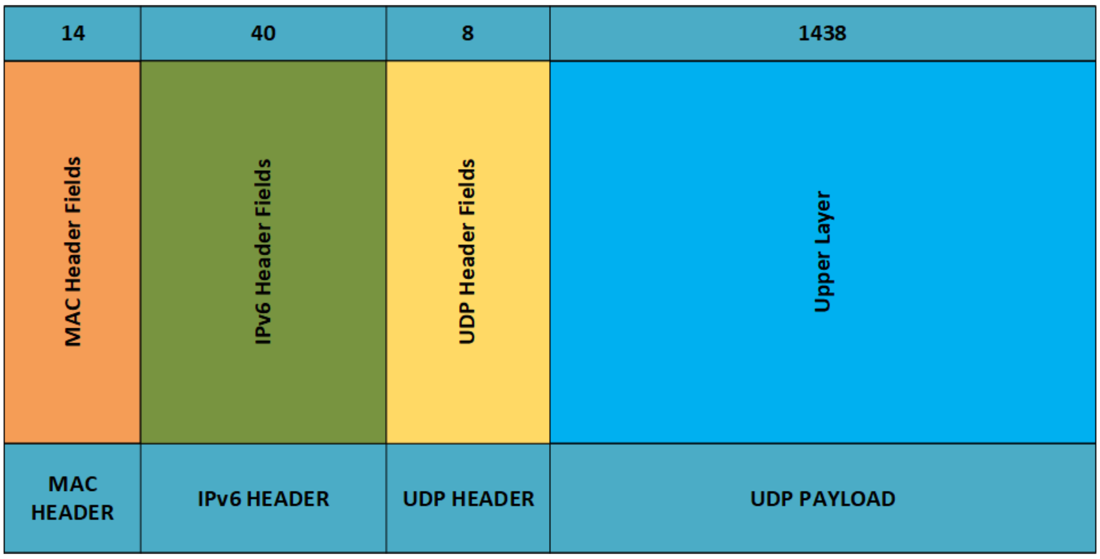


Figure 5.3: Data message packet format over DSRC [12]

The maximum payload size of a Zigbee packet is 66 bytes. We assume that we have temperature sensors that collect 16 readings per hour (every 3 minutes). Each reading is a floating-point of 4 bytes; thus, the resulting payload consumed is 64 bytes after sending the data to the IoT cluster head. It concatenates the unique 8 byte source address found in the Zigbee MAC header along with the 64 bytes of data collected, which adds up to 72 bytes. Each IoT cluster head can service 19 IoTs at a time in each data request, thus consuming 1368 bytes out of the maximum payload size, which is indicated in Figure 5.3. If the IoT coordinator collects data from more than 19 IoTs, it will split the data into multiple requests, however, in our clusters we assume they are groups of 19 IoTs maximum. The sensory readings are offloaded in the aim of applying predictive machine learning algorithms, which is considered as a substantial task to be handled locally by the IoT. The IoT is considered a computationally and energy limited device, which is why it is considered a substantial task. However, to simulate different processing times of the predictive machine learning tasks allocated to the vehicles, we range task computation times from 0 - 5 seconds, which is more than sufficient given the instantaneous output that is typically produced by a trained deep learning model seen in [58, 59]. We assume the OBU of a vehicle posses the minimum specifications mention in Table 5.5 below. It is important to take into consideration that a modern vehicle's OBU is more powerful than the specifications mentioned in Table 5.5.

Microprocessor	2.5 GHz Intel Core i5-3210M
Memory	4 GB 1600 MHz DDR3
Video Graphics	AMD Radeon HD 7670M (2 GB DDR3 dedicated)
Hard Drive	750 GB SATA (5400 rpm)

Table 5.5: OBU minimum specification

In our approach, we take care of the communication between the IoT coordinator and other utility devices such as public vehicles and RSUs. We assume that the clustering mechanism and leader election for IoT devices are background processes and out of the scope of this thesis. We adopted clustering to decrease the amount of traffic sent accompanied by a decrease in interference. The interference decreases since the amount of interference is directly proportional to the number of simultaneous transmissions. Interference causes packets to get dropped due to the attenuations in the received signal at the receiver's end. This could happen to both the IoT and the MCN. On the IoT's end, if an offer message is sent from an MCN, this offer message can get dropped due to the interference and noise. Similarly, on the MCN's end, if an association message is sent from an IoT device, this association message can get dropped due to the interference and noise. Thus, the clustering is crucial for enhancing the communication between IoT coordinators and other utilities. Clustering is also crucial for decreasing the rate of dropped packets.

Our system is composed of many components that interact using DSRC to offload data directly from an IoT to an MCN and retrieve the data later on. Our IoT devices are considered as resource limited.

IoT cluster heads, which possess a task that needs to be offloaded, will wait until they receive advertisements, which are in the form of "hello" messages, from vehicles passing by. Consequently, the IoT head directly after receiving the advertisement message, assesses the SINR and checks if it is above the threshold (7 dB) which is adopted from [55]. If it is above the threshold, the IoT head will immediately send out an association request to the specified vehicle which includes the position of the IoT head. After receiving the association message at the vehicle's end, the MCN will also calculate the SINR and checks if it's above the specified threshold. We assume all cars in our simulation are equipped with a geographical positioning system that allows the vehicle to calculate the expected time of connection with the IoT head on behalf of the IoT head, thus decreasing the computation upon the IoTs. Additionally, the vehicle will check if the approximate time of connection is above the specified threshold indicated in Table 5.2. If the two conditions are met, the automobile will reply with an offer request. This offer request will be received by the IoT head, which holds information about the SINR upon receiving the association message on the vehicle's behalf. After that, it will reply with the data computation request that holds the task information.

It is worth noting that we implement a task buffer on public vehicles, which would increase the availability of resources. Thus, even if a car accepts a task, it will continue broadcasting its services until the buffer is full. Furthermore, the task buffer is sized so that the vehicles OBU is able to serve the IoT devices without being overloaded. Moreover, the clustering of IoT devices and implementing a task buffer improves the scalability and availability of the resources. What is meant by availability here is the ability for one car to service more than one IoT head, which in turn improves the feasibility of our approach in low-density areas. But availability is directly coupled with scalability, where the adopted methods above provide the network with the capacity for unpredictable growth. Moreover, to ensure that we simulate a real-life scenario, we randomized vehicle trips using a random trips generator, a tool provided by SUMO. It generates a set of random trips for a given network. It does so by choosing the source and destination edge. The trips are distributed evenly in an interval defined by the beginning and end time in seconds and the repetition rate represents the number of trips in seconds. Moreover, we added the fringe factor option to increase the probability that trips will start and end at the fringe of the network. If the value were 10, for example, edges with no successor or predecessor would be ten times more likely to be chosen as the starting point or end point of a trip. This feature is useful for modeling through traffic, which starts and ends at the outside of the simulated area.

5.2 Results and Evaluation

To simulate our approach, we spread on average 1 IoT per 1 m^2, which results in 6421 IoTs spread throughout our simulation area. However, as mentioned earlier, we adopted the clustering mechanism to decrease the number of transmitting devices, which reflects positively on the SINR levels [60]. Furthermore, we cluster the 6421 into groups each of 19 devices which we discussed in the previous section. Thus, there are 337 IoT cluster heads on average dispersed randomly across the area.

To calculate the number of runs required for achieving at least a 90 percent confidence level, we ran a scenario with default parameter values ten times. For each simulation run, the pseudo-random generator seed (based on the script process ID), the movement file, and the dispersion of IoT cluster heads were changed. The average successfully retrieved computation ratio and the average serviced IoT cluster heads were calculated. Consequently, the number of runs required to achieve 90 percent confidence was computed using the central limit theorem, as discussed in [61, 62]. The +/- precision value for the ratios mentioned above is 0.03. However, the number of runs was recomputed as we varied the vehicle density from twenty to a hundred cars per hour of simulation. This variation resulted in a different number simulation runs depending on the vehicle density. The ten samples consistently output the mean and standard deviation in close range to each other for a certain vehicle density. The maximum simulation run was 10. On average, we performed 15 simulation runs for each vehicle density.

It is worth mentioning that in our simulation, we assume, during one hour of the simulation, the IoT will offload 16 temperature recordings collected over one hour. Each is a 4-byte floating-point. Thus, every IoT will offload 64 bytes of data within the 66 bytes limited payload size of a Zigbee packet we discussed in the previous section. After the IoT cluster head receives the data from the cluster members, it will concatenate the 8 byte source address of each cluster member shown in Figure 5.2 to their respective Zigbee payload. This concatenation will lead to accumulating 72 bytes from each cluster member per hour. We assume each cluster is made of 19 participants, including the cluster head. Therefore, 1368 bytes of data will be accumulated from the whole group. The total size of the data is less than the DSRC maximum payload size as indicated in Figure 5.3.

5.2.1 Varying queue size (1 - 50)

To study the effect of implementing a task buffer on MCNs we varied the queue size from 50 tasks to 1 task at a time where the tasks were done sequentially in FIFO order. We fixed the number of cars and IoT devices (37 vehicles and 337 IoT cluster heads). It worth mentioning each MCN had a random path. This randomness helps to simulate a real-life scenario. Our framework recorded a retrieval rate (number of IoT cluster heads offloaded data and retrieved out of all the IoT cluster heads offloaded) of more than 90%. This shows the efficiency of our selection mechanism, which depends on the SINR levels and TOC. We were able to decrease the number of packet drops caused by inadequate environmental circumstances such as interference and the mobility of the vehicles.

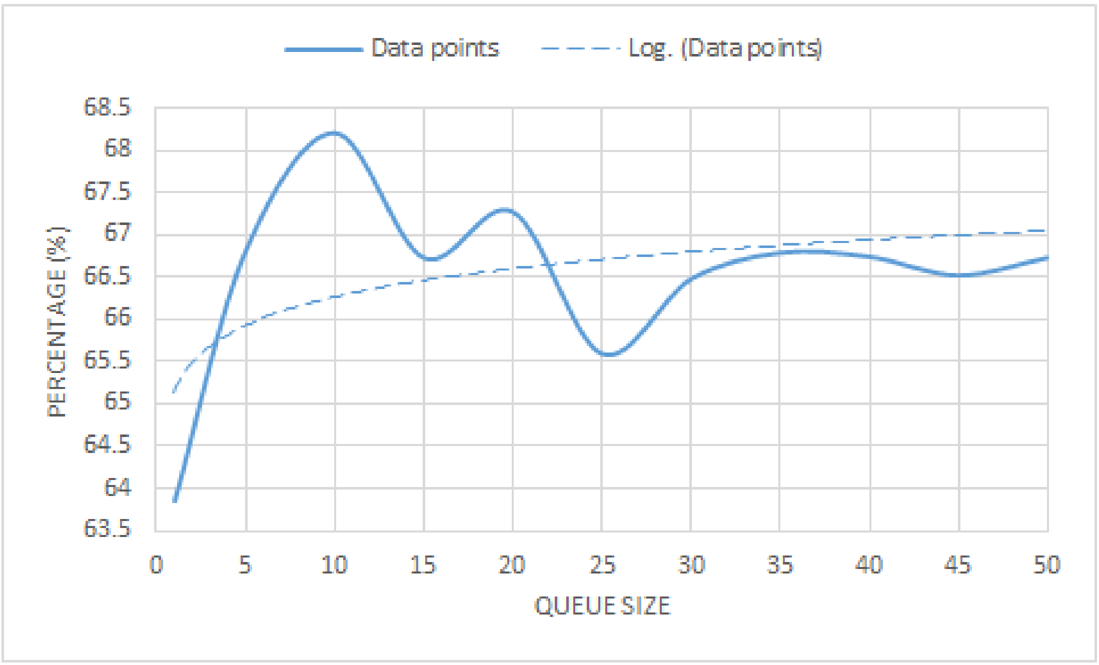


Figure 5.4: Percentage of serviced IoT cluster heads while varying vehicle's queue size

Moreover, the effect of varying the queue size on the availability of resources was studied. The recorded results are shown in Figure 5.4, which shows a monotonic increase in the percentage of serviced IoT cluster heads (number of IoT cluster heads that offloaded and retrieved their data out of the total number of deployed IoT cluster heads). This increase shows how the implementation of a car task buffer affects the availability of resources to service IoT devices positively, which increases the availability of resources when they are inherently limited.

5.2.2 Varying vehicle density (20-100) while varying queue size (50 - 1)

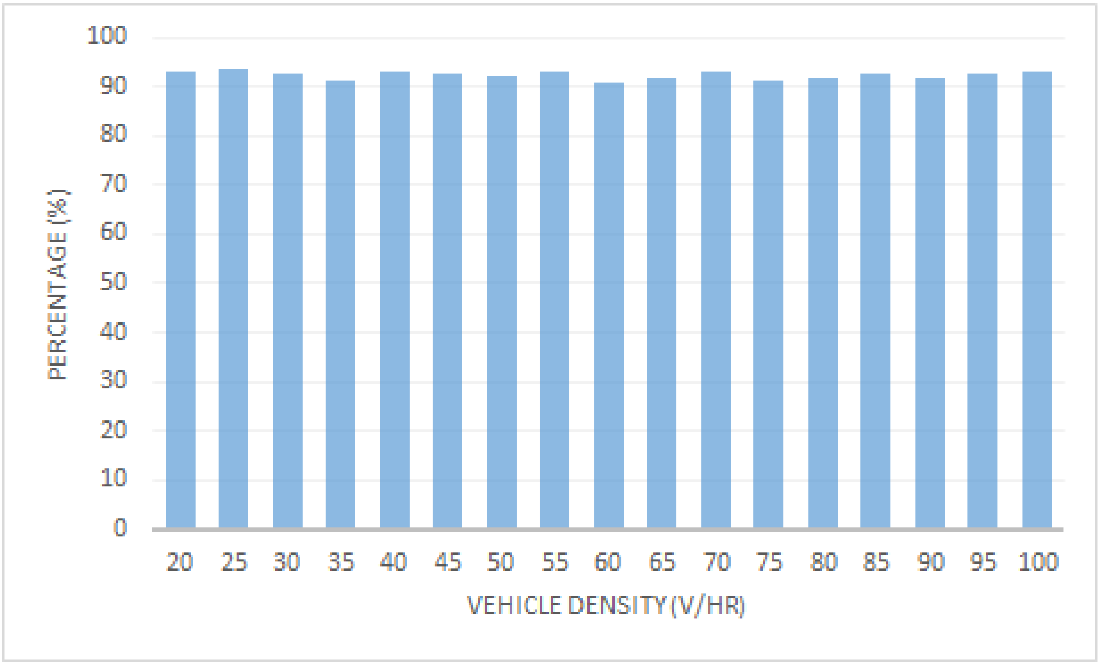


Figure 5.5: Successful retrieval rate while varying vehicle density

To study the feasibility of our proposed model, we study the successful retrieval rate of tasks. As discussed earlier, we varied the vehicle density from 20 cars per hour to 100 car per hour while changing the route of every vehicle and altering the spread of IoT cluster heads over the respective area randomly. The successful retrieval rate ranges between 90% to 95%. The high retrieval rate reflects on the feasibility of our proposed model. These results align with our prior findings where we varied the queue size which show the efficiency of the selection mechanism described previously in section 4.2.3. Minor fluctuations in the retrieval rate depicted in Figure 5.5 which caused a decrease in the successful retrieval rate are due to the dropping of packets caused by interference from neighboring devices. However, some loss is due to the absence of an available vehicle to return results to the IoT cluster head from the RSU network. This behavior depends on the opportunistic behavior of VANETs. Furthermore, the successful retrieval rate shows that 5% to 10% of the results are lost.

The high retrieval rate over different car densities shows that the successful retrieval rate is not dependent on vehicle density. It does however depend on the opportunistic availability of vehicles and the strategic MCN selection adopted.

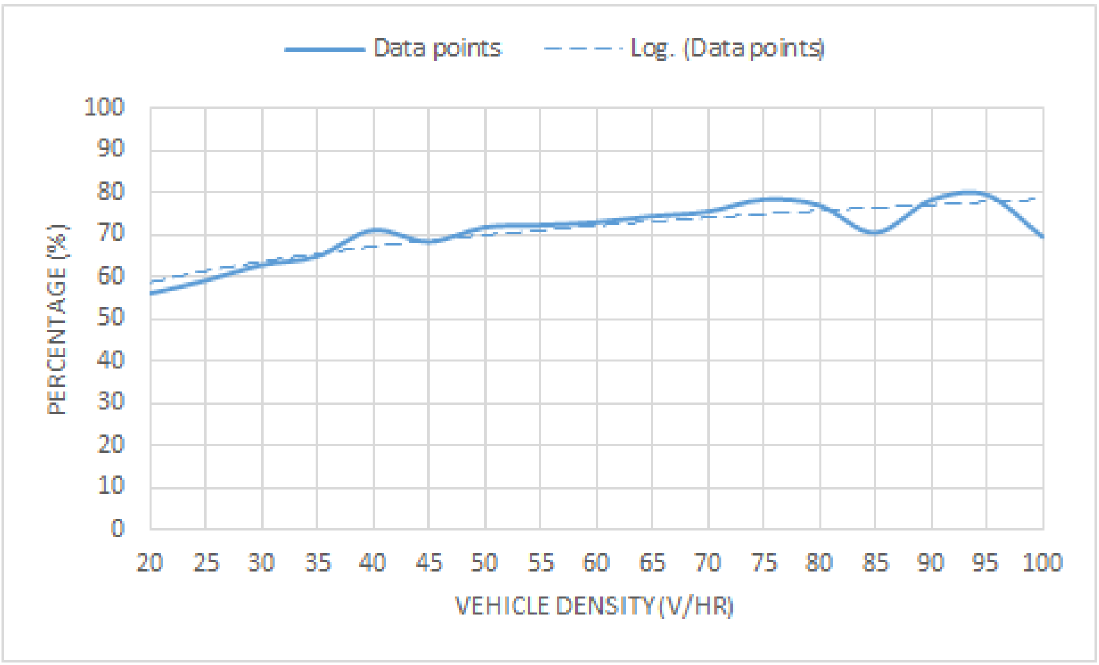


Figure 5.6: Percentage of serviced IoT cluster heads while varying vehicle density

The percentage of serviced IoTs is depicted in Figure 5.6 above. A critical factor to study is the percentage of IoTs serviced as we vary the vehicle density while altering the spread of the 337 IoT cluster heads randomly. The percentage of serviced IoTs range from 58% up to 80%, which conveys the direct relationship between the rate of serviced IoTs and vehicle density. Figure 5.6 shows an increase in the percentage of serviced IoTs as we increase the vehicle density. Due to the randomized vehicle routes, fluctuations occur, causing these variations in the serviced rate seen in Figure 5.6. Furthermore, the trend is conserved where increasing vehicle density increases the percentage of serviced IoTs. We indicate that serviced IoTs is the percentage of vehicles that successfully offloaded and retrieved their tasks out of the total number of IoTs. As vehicle density increases, we expect an increase in serviced IoTs. This relationship between the vehicle density and the serviced IoTs is validated through the results obtained where the logarithmic trend line shows the expected behavior while increasing the density.

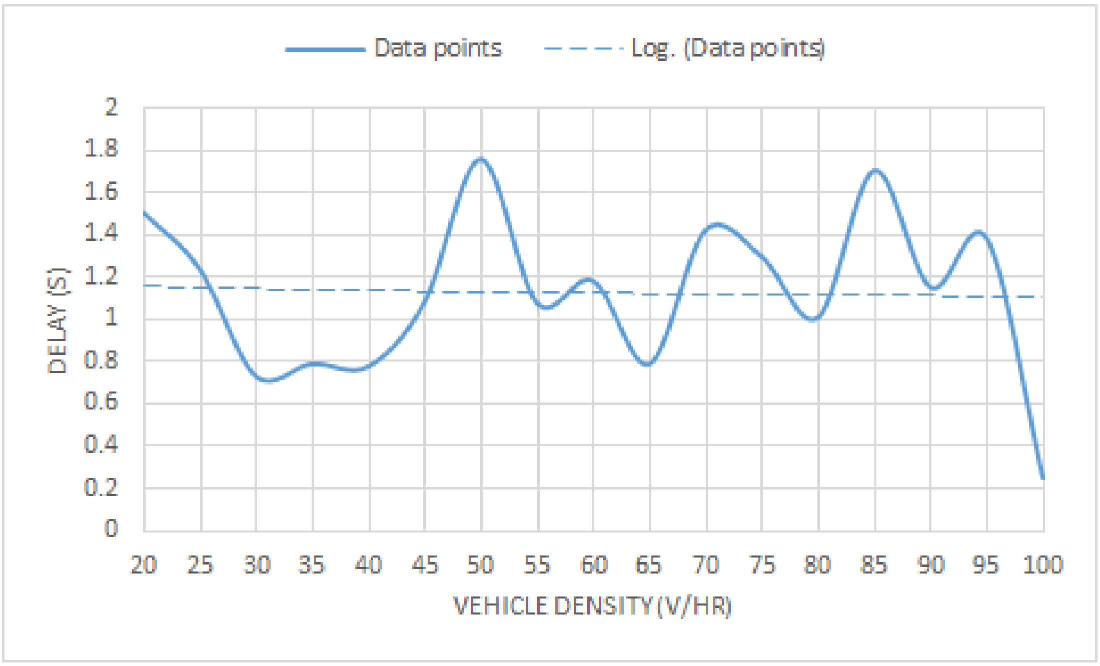


Figure 5.7: Retrieval delay vs vehicle density

Furthermore, we study the data retrieval delay which is the wait time of IoT devices without considering task time to retrieve its results. The overall average wait time is 1.12 seconds. These results are considered as a positive indication of the applicability of this approach in real life, although compared to the cloud, it is notable somehow. A trade-off exists between the use of idle resources with a little bit of insignificant delay and the congestion incurred by overwhelming the LTE infrastructure. Moreover, as depicted in Figure 5.7, the variations are attributed to the various possible combinations of roads a vehicle can take as soon as it enters the simulation, which we indicate by randomness of the trips.

However, as the vehicle density increases, the trend of the service delay is decreasing due to the availability of suitable MCNs passing by. In addition, the controlled fluctuation, which is not considered as severe, contributes to the usability and applicability of such a system. We can note that the trend shown in Figure 5.7 showing a decreasing inclination as we increase the number of vehicles is reasonable. The curve above shows the time delay for IoTs, which includes the computation time. These trends show the gradual decreasing trend over increasing the vehicle's density.

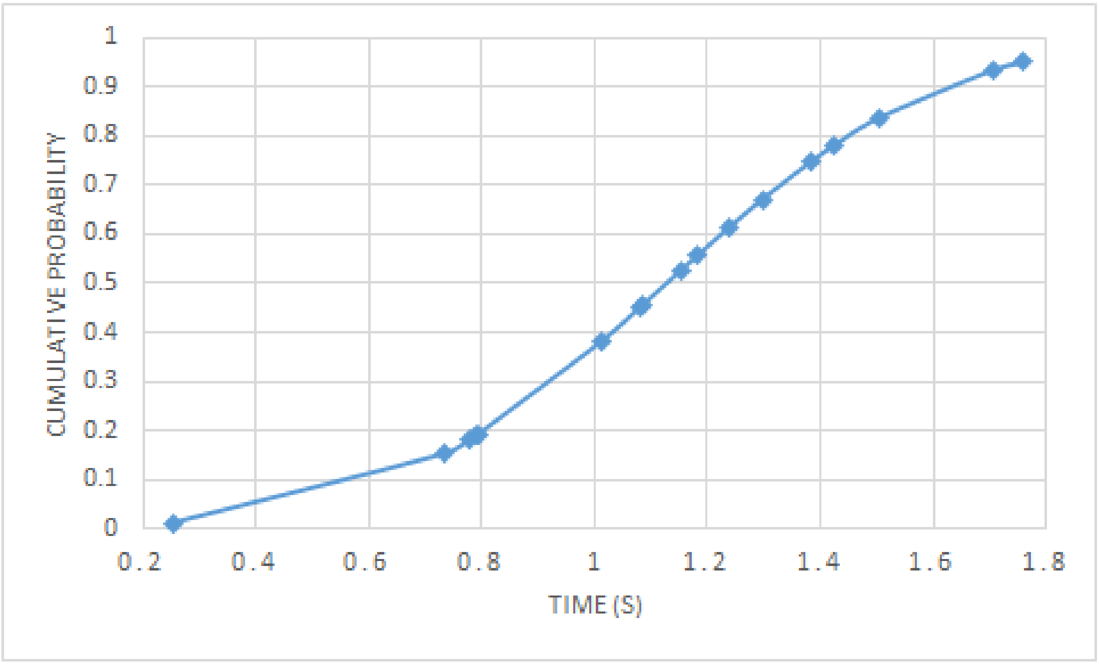


Figure 5.8: CDF retrieval delay of IoT

The mean 1.12 seconds, almost 70% of the distribution lies one standard deviation away (0.38). This is depicted above in Figure 5.8. It can be noticed that most of the data points are concentrated around the mean +\− 0.38 (0.74 - 1.5), where on average, an IoT must wait 0.4 seconds before retrieving the result after offloading the task to an MCN taking into consideration task time and 1.12 seconds. However, this value is expected to be lower as the number of vehicles is increased. This relationship exists since vehicles with better conditions might be available and service the IoT cluster head needs.

5.2.3 Varying IoT density (337 - 17)

Another critical factor in our study is scalability. To investigate this factor at first, we study the system performance with decreased resources (less vehicle density) of only 35 vehicles each with a queue size of 10. Our system's performance with limited resources reveals our system's necessity and usability in real-life scenarios where resources are scarce, and the importance and necessity of the proposed system prevails. To address this crucial factor, we vary the IoT density from 17 to 337 and study the percentage of tasks retrieved successfully along with the rate of serviced IoTs cluster heads, as depicted in the figures 5.9 and 5.11 and the retrieval delay as depicted in Figure 5.10.

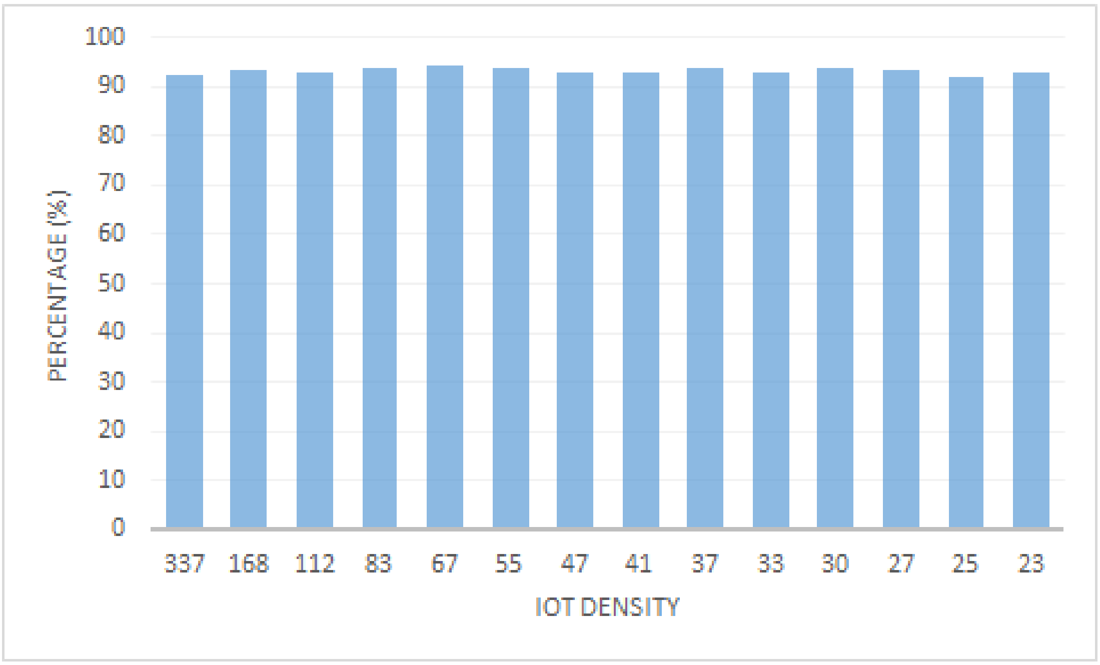


Figure 5.9: Successful retrieval rate while varying IoT density

We obtain results with at least 90% confidence interval as suggested by [61, 62]. Furthermore, the results that showed a consistent mean and standard deviation over several runs are at least 90% close to the true mean of the distribution. We adopted this model due to the time it takes for ns-3 and SUMO to simulate a scenario.

Figure 5.9 shows a successful retrieval rate above 90% for all vehicle densities, which shows that for lower frequencies, the retrieval rate maintains a minimum level of almost 90%. This clearly shows the selection strategy adopted based on the TOC and SINR levels is an efficient mechanism. Practically 90% of the IoT cluster heads who offloaded got their results, and only 10% lost their computation due to propagation loss or the unavailability of vehicles that can retrieve data from the RSU network.

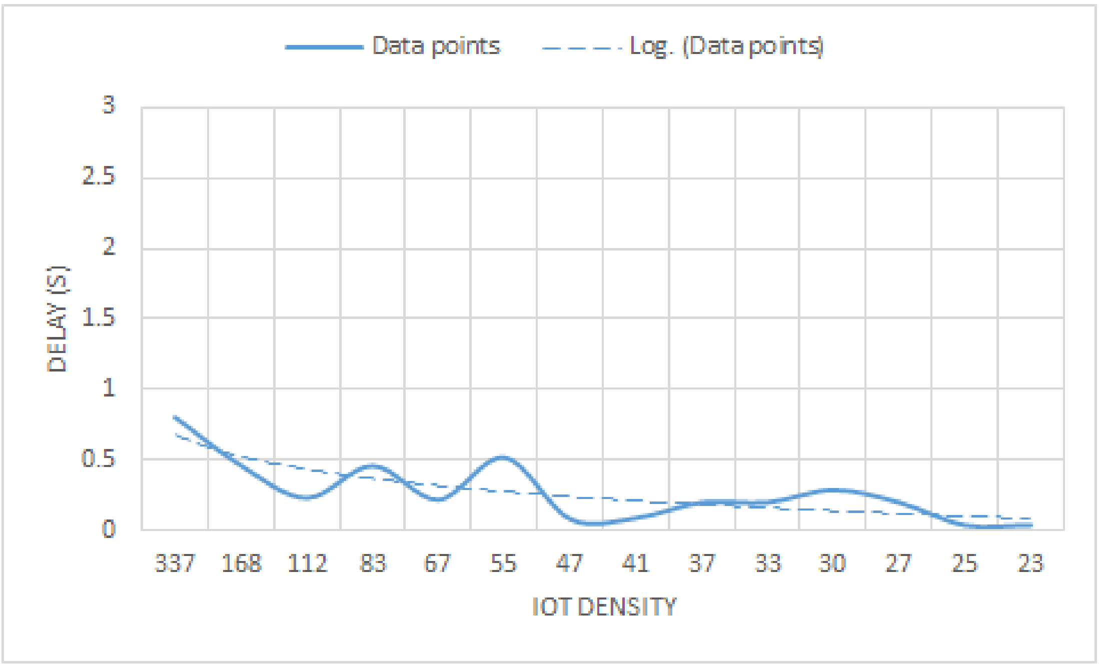


Figure 5.10: Retrieval delay while varying IoT density

Furthermore, in Figure 5.10, we display the retrieval delay to a decreased vehicle density, which shows decrease in the delay as we decrease the IoT density. And that is due to the decrease in the number of simultaneous requests of IoT devices, since the request number is directly proportional to the IoT density.

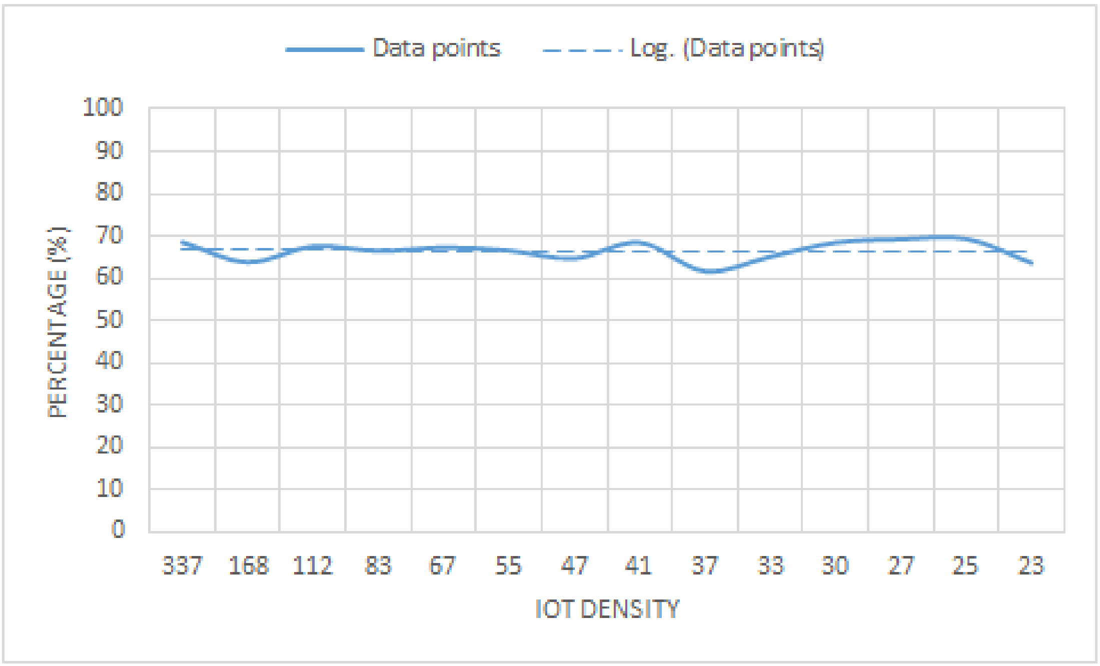


Figure 5.11: Serviced rate as we vary IoT density

While having a vehicle density of 35 per hour is relatively low compared to the area size, we plot the percentage of serviced IoTs with respect to different densities shown in Figure 5.11. The conveyed results show the preservation of a level of at least 60%. Having a percentage higher than 60% with low vehicle density (35 vehicle\hr) also contributes to the proof of scalability of our system. It is worth noting that the percentage of serviced IoTs is relatively low due to the randomness of the vehicle's path. This shows the ability of 35 vehicles to serve increasing IoT densities. These results show the increased availability of resources where the number of vehicles can serve approximately three times their density.

The results are classified to 3 categories.

1. The data retrieved from the vehicle directly.
2. Vehicle relayed the result to the RSU network and the RSU network sent the result to the IoT.
3. The results were relayed back using a third vehicle where the RSU relayed the result to it so it gives it back to the IoT.

The average percentage of the retrieved tasks classified according to type is depicted in Table 5.6 below:

Variation type	Type 1	Type 2	Type 3
Queue Size	95%	1.5%	3.5%
Vehicle Density	98%	0.5%	1.5%
IoT Density	99%	0.5%	0.5%

Table 5.6: Average percentage of retrieval type

The results in Table 5.6 show that most of the tasks are of the first type and that is due to the conditions set for the selection mechanism. The selection mechanism as mentioned earlier will aid in finding the best candidate to be able to fully satisfy the computation request by the vehicle itself with the assistance of the physical infrastructure. Thus, keeping the load on the physical infrastructure as low as possible.

5.2.4 Stress Testing

In line with our previous experiments, we also worked on testing the limits of our framework. The experiments are documented the below subsections.

5.2.4.1 Vary Request Rate

We increased the request rate where each IoT device would request a task from the IoT cluster head. In one task the IoT device will record 16 temperature data points combined, which will be relayed to the IoT cluster head, in turn, will offload the collected tasks from all the IoT devices to passing by MCNs. The setup of the experiment is summarized in the Table 5.7 below.

Queue Size	20
MCN density	50
MCN Processing Time (s)	0.1
IoT Cluster Head Density	337
IoT Density	6403

Table 5.7: Varying request rate experimental setup

To test the limits of our framework, we varied the request rate from 1 request per second per IoT device to 1 request per hour per IoT device. According to the request rate, the IoT device chooses randomly to send a request. If it already sent a request and did not retrieve the result back the IoT will be sending the same request again if it randomly decides to send a request again and these requests are what we call delayed requests. However, if it did not send a request earlier it will be generating new request which we call original request. The results are depicted in Figure 5.12

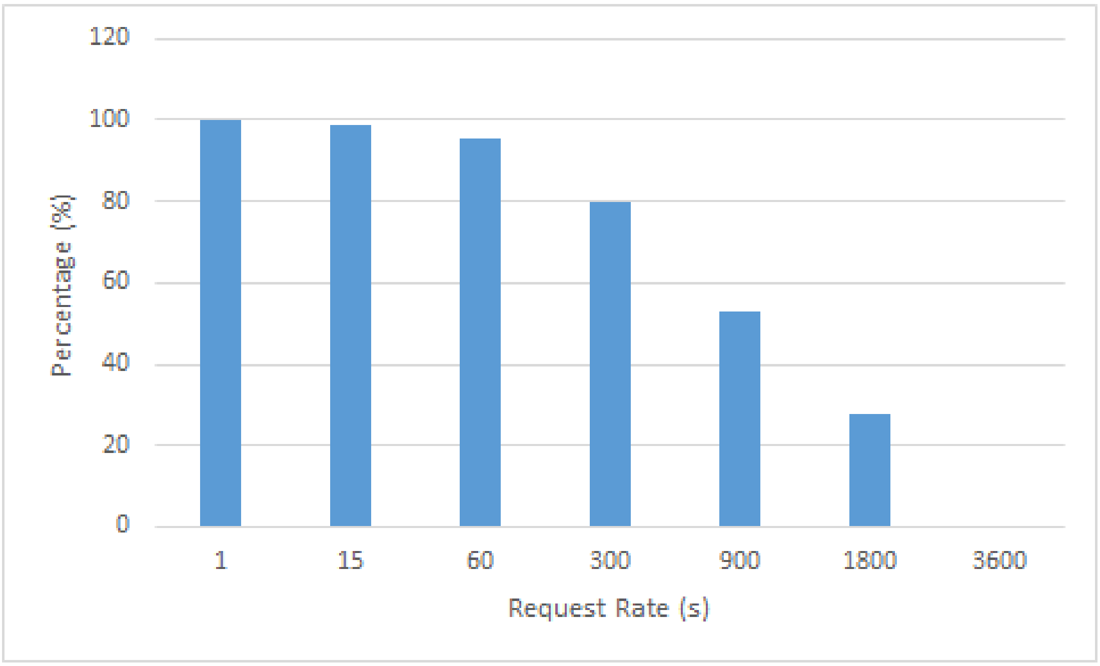


Figure 5.12: Percentage of delayed requests as we vary the request rate

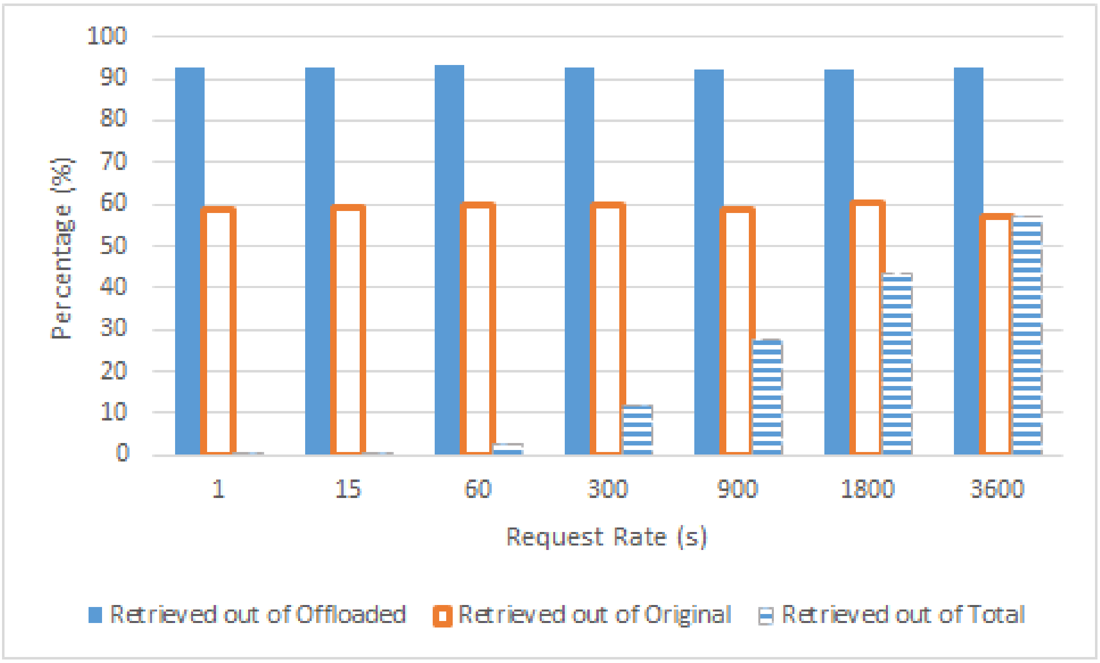


Figure 5.13: Percentage of retrieved requests as we vary the request rate

Figure 5.12 shows the variation of the percentage of delayed requests while varying the request rate. As depicted, the percentage of delayed requests is high when setting the request rate to 1 and 15 seconds these results are reasonable since we have a limited number of vehicles and a vast area. It is worth mentioning that request rates 1 and 15 are extreme cases that are used for the sole purpose of stress testing the framework. However, the percentage of delayed requests decreases to 0 as we decrease the rate to 1 task request per hour per IoT device.

Moreover, Figure 5.13 shows the variation of the retrieved requests out of the offloaded, original, and the total number of requests. Offloaded requests are the requests that are offloaded from the IoT cluster head to the MCN.

Request Rate (s)	Average Number of Original Requests
1	17634.9
15	17618.1
60	17283.2
300	15468.5
900	12067.7
1800	9262.5
3600	6403

Table 5.8: Average number of original requests while varying request rate

The percentage of retrieved results out of the offloaded tasks shows a high percentage (above 90%) as we vary the request rate. This aligns with our previous findings in the sections above. It shows a high retrieval rate which is attributed to the selection mechanism of our framework. Furthermore, the average percentage of retrieved results out of the original number of requests are depicted in Table 5.8, which shows the direct coupling of the decrease in the request rate and the decrease of original requests. The percentage of retrieved requests out of the original requests ranges between (55% - 60%), which shows positive results for high request rates where the system was stressed by increasing the request rate tremendously while keeping in mind the limited number of MCN with a relatively small queue size.

Request Rate (s)	Average Number of Total Requests
1	23044397
15	1536720
60	384180
300	76836
900	25612
1800	12806
3600	6403

Table 5.9: Average number of total requests while varying request rate

The variation of the request rate increases the number of total requests as we increase the request rate as shown in Table 5.9. The percentage of retrieved requests out of the total number of requests increases monotonically as we decrease the requests rate as depicted in Figure 5.13. This increase is attributed to the decrease in the total number of requests. The results also show that our framework is able to serve IoT environments in extreme conditions (50 vehicles per hour with limited queue size of 20 and a request rate every 1 and 15 second).

5.2.4.2 Vary Queue Size

To test the effect of the queue size on our framework. We varied the queue size from 20 to 250 while increasing the request rate to 1 task per 60 s per IoT device. While stabilizing the vehicle density and IoT cluster head. The setup of the experiment is summarized in the Table 5.10 below.

Request rate (s)	60
MCN density	50
MCN Processing Time (s)	0.1
IoT Cluster Head Density	337
IoT Density	6403

Table 5.10: Varying queue size experimental setup

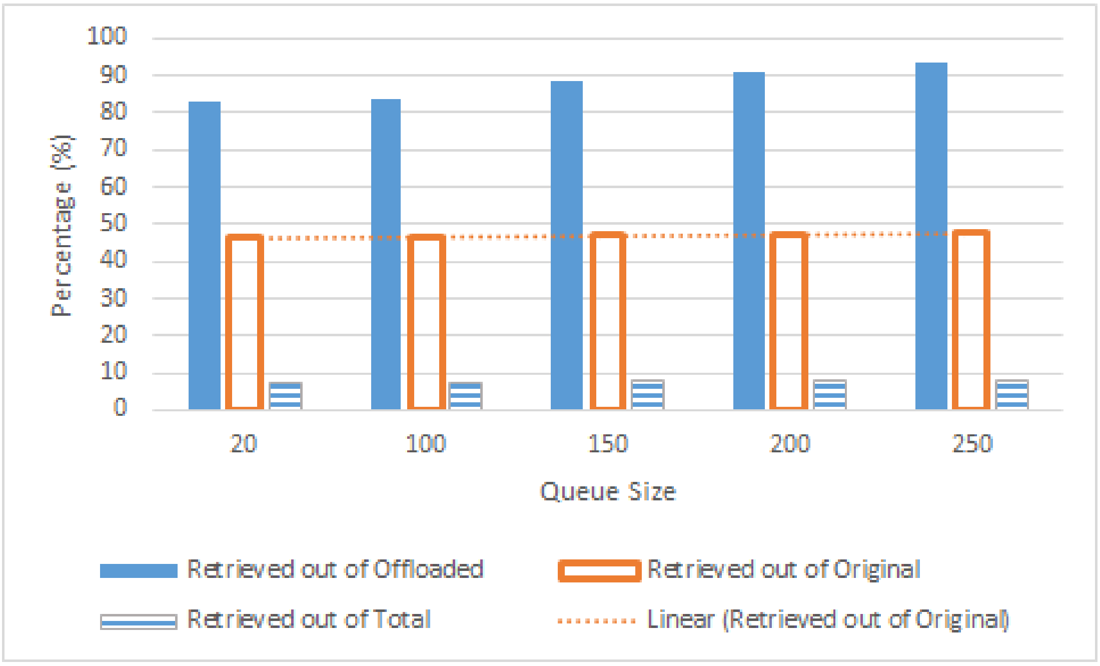


Figure 5.14: Percentage of retrieved requests as we vary queue size

The results that are depicted above in Figure 5.14 show an enhancement in the percentage of retrieved results. Where the percentage of the retrieved out of the categories mention in the figure show an increase in the percentage which depicts the positive effect of the queue on the our framework. This shows the performance of the proposed system under extreme conditions and with a limited number of vehicles.

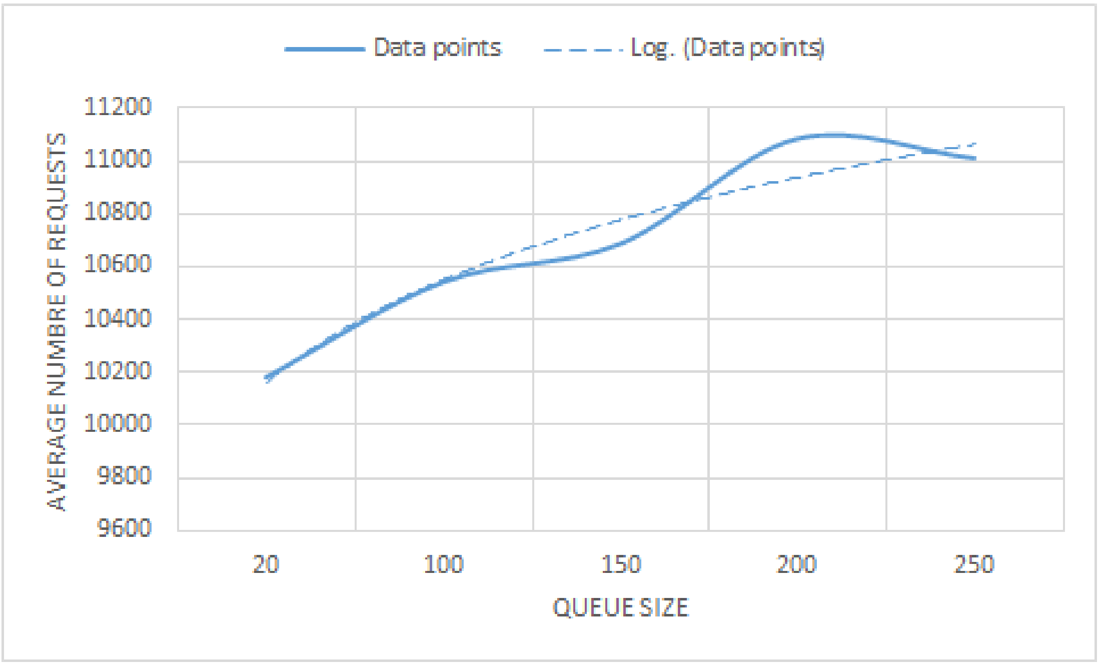


Figure 5.15: Average number of offloaded requests as we vary queue size

In Figure 5.15 the number of offloaded requests by the IoTs is increasing and this is attributed to the increase in the queue size which increases the availability of resources when the number of resources is limited to 50 vehicles in the vast area of downtown Ottawa. This supports our claims that the existence of a queue enhance the availability of our framework.

5.2.4.3 Time Delay

In order to check the delay incurred by the communications in our framework we varied vehicle processing time randomly between 10 ms to 30,000 ms (30 s) while keeping the vehicle and IoT density constant, with an increased request rate. The experiment's setup is summarized below in Table 5.11.

Request rate (s)	60
MCN density	50
MCN Processing Time (ms)	10 - 30,000
IoT Cluster Head Density	337
IoT Density	6403

Table 5.11: Time delay experimental setup

The results are depicted in the Table 5.12 shows the time delay in seconds according to each type of retrieval which was described in section 5.2.3.

Briefly the three categories are:

1. The data retrieved from the vehicle directly.

2. Vehicle relayed the result to the RSU network and the RSU network sent the result to the IoT.

3. The results were relayed back using a third vehicle where the RSU relayed the result to it so it gives it back to the IoT.

	Average time delay per Category (s)	Percentage per category (%)
Category 1	12.45	80.10
Category 2	141.94	16.26
Category 3	951.78	2.64

Table 5.12: Time delay results

The results show that the time delay for category 2 and 3 are larger from that of type 1 which is reasonable since the 2^{nd} category incurs some delay from finding the RSU network in the vicinity of the MCN. Whereas 3^{rd} category incurred delay similar to 2^{nd} category and delay at the RSU network to find a suitable MCN to relay the results back to the IoT which depends on the randomness of vehicles trajectory. It is worth mentioning that the average overall delay is 58.3452 seconds with an average of 11 queued tasks per MCN. It is worth noting that the delay shown over here include the processing time at the vehicle.

5.2.5 Comparative Study

Briefly, to efficiently retrieve offloaded tasks from vehicle and RSUs as relaying entities, [1] proposed a time constrained retrieval protocol. Vehicle will offload their data to a broker RSU which will handle offloading the tasks to other vehicles in the vicinity. However, in our approach we are using IoT devices with limited power supply which would decrease the transmission power. This affects the throughput and increase the loss where the signal undergo some attenuation due to environmental and other interfering signals. Their dependence on a broke affects the performance of the system along with their lack for a selection strategy.

To compare our work to theirs we simulated a similar environment, however, with reduced vehicle density of 20 vehicle per hour over the whole area. The results are depicted below in the Table.

	Average retrieval percentage (%)
[1]	42
Our approach	90

Table 5.13: Comparative table

In [1], the average retrieval percentage for tasks of 0 to 10 seconds is 42% where the results were one hop away, whereas our retrieval rate for 1 hop away tasks is much higher and that is attributed to our effective selection mechanism and the direct communication between the entities without a broker in the middle. In [1] when the broker RSU assigns a task to a vehicle it might select a very far vehicle which will rapidly leave the area thus it affects returning the results back to the RSU. However, in our case the IoT devices transmission range is small with respect to that of RSU, this allows the MCN even if it left the area shortly after the handover from the IoT device to be able to deliver back the result since the MCN-IoT communication range is much greater than that of the IoT-MCN communication range.

Chapter 6

Conclusion and Future Work

The Fourth Industrial Revolution (Industry 4.0) is the main driver behind the digitization of enterprises and governmental institutions. Machines, sensors, and actuators can communicate and process data to enable a more efficient production cycle. The Industrial Revolution and the increased proliferation of data by IoTs provoked the incorporation of offloading schemes to help ease the burden upon the IoT devices which suffer from limited computation ability and power supply.

In this thesis we have proposed an opportunistic vehicle based computing framework for IoT offloading. A framework that defines the means of communication between IoT and a mobile vehicular node in a heterogeneous environment, which allows the IoT to offload computation to a nearby vehicle and retrieve the result back. In our approach IoT devices are able to directly communicate with vehicles and utilize their idle resources, this allows the auto-arrangement of IoT devices as clusters and proposed a selection strategy that depends on the SINR and the time needed for offloading data and retrieving the results. Also, we incorporate the decentralization of decision making instead of assigning a device the role of a coordinator and overwhelming it

.

The existence of traditional offloading paradigms (cloud - fog) which suffer from latency and high installation cost motivated the transition to a new paradigm which is mobile computing clouds which can be opportunistically utilizes MCNs to service some tasks from close by IoTs. MCNs which in most cases are public vehicles roaming around the city.

Our proposed framework presents a reliable and a sustainable replacement to traditional paradigm and the surveyed related work. The selection mechanism studied in this work along with the framework that is put in place shows promising results in terms of usability. This framework is adaptable and can be used for different systems not necessarily RSUs but also can be used by other stationary fog nodes to share the load.

In the meantime, our future work will focus on enhancing the RSU forwarding mechanism which will be based on machine learning models that will be gathered by the RSUs from their environment. Moreover, we will work on applying new scheduling mechanisms at the MCN as suggested in [63, 64]. Finally, implementing the framework on real life test beds to assess the how it will perform in real life.

Appendix A

Abbreviations

5G	Fifth Generation
D2D	Device-to-Device
DAG	Directed Acyclic Graph
DSRC	Dedicated Short Range Communication
GCC	GNU Compiler Collection
GPS	Global Positioning System
IoT	Internet of Things
LCB	Local Community Broker
LR-WPAN	Low-Rate Wireless Personal Area Networks
LTE	Long Term Evolution
MCN	Mobile Computing Node
MTU	Maximum Transmission Unit
NS-3	Network Simulator 3
OBU	On-Board Unit
RSU	Road Side Unit
SINR	Signal Interference Noise Ratio
SUMO	Simulation of Urban MObility
TOC	Time Of Connection
TPM	Trusted Platform Module
TTP	Trusted Third Party Units
VANET	Vehicular Ad Hoc Network
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to Everything
WPAN	Wireless Personal Area Network

Bibliography

[1] V. Soto, R. E. De Grande, and A. Boukerche, "Repro: time-constrained data retrieval for edge offloading in vehicular clouds," in *Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, pp. 47–54, 2017.

[2] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling," *IEEE vehicular technology magazine*, vol. 14, no. 1, pp. 28–36, 2018.

[3] K. Mershad and H. Artail, "Finding a star in a vehicular cloud," *IEEE Intelligent transportation systems magazine*, vol. 5, no. 2, pp. 55–68, 2013.

[4] B. Li, Y. Pei, H. Wu, Z. Liu, and H. Liu, "Computation offloading management for vehicular ad hoc cloud," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 728–739, Springer, 2014.

[5] S. Bitam, A. Mellouk, and S. Zeadally, "Vanet-cloud: a generic cloud computing model for vehicular ad hoc networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96–102, 2015.

[6] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog following me: Latency and quality balanced task allocation in vehicular fog computing," in *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, IEEE, 2018.

[7] L. Vigneri, T. Spyropoulos, and C. Barakat, "Storage on wheels: Offloading popular contents through a vehicular cloud," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–9, IEEE, 2016.

[8] N. S. Safa, C. Maple, M. Haghparast, T. Watson, and M. Dianati, "An opportunistic resource management model to overcome resource-constraint in the internet of things," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 8, p. e5014, 2019.

[9] N. Fernando, S. W. Loke, I. Avazpour, F.-F. Chen, A. B. Abkenar, and A. Ibrahim, “Opportunistic fog for iot: Challenges and opportunities,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8897–8910, 2019.

[10] R.-I. Ciobanu, C. Dobre, M. Bălănescu, and G. Suciu, “Data and task offloading in collaborative mobile fog-based networks,” *IEEE Access*, vol. 7, pp. 104405–104422, 2019.

[11] M. Franceschinis, C. Pastrone, M. A. Spirito, and C. Borean, “On the performance of zigbee pro and zigbee ip in ieee 802.15. 4 networks,” in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 83–88, IEEE, 2013.

[12] X. Jiang, X. Cao, and D. H. Du, “Multihop transmission and retransmission measurement of real-time video streaming over dsrc devices,” in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pp. 1–9, IEEE, 2014.

[13] A. Boukerche and E. Robson, “Vehicular cloud computing: Architectures, applications, and mobility,” *Computer networks*, vol. 135, pp. 171–189, 2018.

[14] F. Samie, L. Bauer, and J. Henkel, “Iot technologies for embedded computing: A survey,” in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 1–10, IEEE, 2016.

[15] L. Da Xu, W. He, and S. Li, “Internet of things in industries: A survey,” *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[16] A. R. Biswas and R. Giaffreda, “Iot and cloud convergence: Opportunities and challenges,” in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pp. 375–376, IEEE, 2014.

[17] H. Flores, X. Su, V. Kostakos, A. Y. Ding, P. Nurmi, S. Tarkoma, P. Hui, and Y. Li, “Large-scale offloading in the internet of things,” in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 479–484, IEEE, 2017.

[18] K. Akherfi, M. Gerndt, and H. Harroud, “Mobile cloud computing for computation offloading: Issues and challenges,” *Applied computing and informatics*, vol. 14, no. 1, pp. 1–16, 2018.

[19] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “Clonecloud: elastic execution between mobile device and cloud,” in *Proceedings of the sixth conference on Computer systems*, pp. 301–314, 2011.

[20] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pp. 49–62, 2010.

[21] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *International Conference on Mobile Computing, Applications, and Services*, pp. 59–79, Springer, 2010.

[22] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, pp. 37–42, 2015.

[23] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

[24] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 325–329, IEEE, 2014.

[25] S. Mubeen, P. Nikolaidis, A. Didic, H. Pei-Breivold, K. Sandström, and M. Behnam, "Delay mitigation in offloaded cloud controllers in industrial iot," *IEEE Access*, vol. 5, pp. 4418–4430, 2017.

[26] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-based computing and storage offloading for data synchronization in iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4272–4282, 2018.

[27] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer applications*, vol. 40, pp. 325–344, 2014.

[28] M. Gerla, "Vehicular cloud computing," in *2012 The 11th annual mediterranean ad hoc networking workshop (Med-Hoc-Net)*, pp. 152–155, IEEE, 2012.

[29] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a resource (vaar)," *IEEE Network*, vol. 29, no. 1, pp. 12–17, 2015.

[30] A. Ashok, P. Steenkiste, and F. Bai, "Vehicular cloud computing through dynamic computation offloading," *Computer Communications*, vol. 120, pp. 125–137, 2018.

[31] J. Zhang, "A survey on trust management for vanets," in *2011 IEEE International Conference on Advanced Information Networking and Applications*, pp. 105–112, IEEE, 2011.

[32] F. Li and Y. Wang, "Routing in vehicular ad hoc networks: A survey," *IEEE Vehicular technology magazine*, vol. 2, no. 2, pp. 12–22, 2007.

[33] M. N. Mejri, J. Ben-Othman, and M. Hamdi, "Survey on vanet security challenges and possible cryptographic solutions," *Vehicular Communications*, vol. 1, no. 2, pp. 53–66, 2014.

[34] H. Hartenstein and L. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Communications magazine*, vol. 46, no. 6, pp. 164–171, 2008.

[35] D. Jiang and L. Delgrossi, "Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments," in *VTC Spring 2008-IEEE Vehicular Technology Conference*, pp. 2036–2040, IEEE, 2008.

[36] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, "The case for cyber foraging," in *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pp. 87–92, 2002.

[37] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power iot edge devices," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 7–12, IEEE, 2016.

[38] K. K. Patel, S. M. Patel, *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[39] G. Gardašević, M. Veletić, N. Maletić, D. Vasiljević, I. Radusinović, S. Tomović, and M. Radonjić, "The iot architectural framework, design issues and application domains," *Wireless personal communications*, vol. 92, no. 1, pp. 127–148, 2017.

[40] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[41] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and devices for the 5g era," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 90–96, 2014.

[42] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, “Mobility-aware caching and computation offloading in 5g ultra-dense cellular networks,” *Sensors*, vol. 16, no. 7, p. 974, 2016.

[43] Q. Zhang, C. Zhu, L. T. Yang, Z. Chen, L. Zhao, and P. Li, “An incremental cfs algorithm for clustering large data in industrial internet of things,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1193–1201, 2017.

[44] L. Xu, R. Collier, and G. M. OHare, “A survey of clustering techniques in wsns and consideration of the challenges of applying such to 5g iot scenarios,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1229–1249, 2017.

[45] L. Ding, P. Shi, and B. Liu, “The clustering of internet, internet of things and social network,” in *2010 Third International Symposium on Knowledge Acquisition and Modeling*, pp. 417–420, IEEE, 2010.

[46] X. Shao, C. Yang, D. Chen, N. Zhao, and F. R. Yu, “Dynamic iot device clustering and energy management with hybrid noma systems,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4622–4630, 2018.

[47] S. S. L. Preeth, R. Dhanalakshmi, R. Kumar, and P. M. Shakeel, “An adaptive fuzzy rule based energy efficient clustering and immune-inspired routing protocol for wsn-assisted iot system,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–13, 2018.

[48] M. Charaf, H. Artail, and Y. Nasser, “Mobile relay node in public transportation for serving outside lte cell edge users,” in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 59–66, IEEE, 2015.

[49] K. Sarieddine, M. Charaf, M. Ayad, and H. Artail, “A framework for mobile relay node selection for serving outdoor cell edge users,” *Computer Networks*, p. 107359, 2020.

[50] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang, “Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks,” in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 288–294, IEEE, 2016.

[51] nsnam, “ns-3 simulator.” `https://nsnam.org`, Feb. 2019.

[52] Eclipse, “Sumo Traffic Simulator.” `https://www.eclipse.org/sumo`, Feb. 2019.

[53] P. Schmager and S. Kuhlmorgen, “ns3-sumo-couplingl.” `https://github.com/vodafone-chair/ns3-sumo-coupling.git`, Feb. 2019.

[54] J. Benin, M. Nowatkowski, and H. Owen, "Vehicular network simulation propagation loss model parameter standardization in ns-3 and beyond," in *2012 Proceedings of IEEE Southeastcon*, pp. 1–5, IEEE, 2012.

[55] Z. Tong, H. Lu, M. Haenggi, and C. Poellabauer, "A stochastic geometry approach to the modeling of dsrc for vehicular safety communication," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1448–1458, 2016.

[56] P. Li, J. Li, L. Nie, and B. Wang, "Research and application of zigbee protocol stack," in *2010 International Conference on Measuring Technology and Mechatronics Automation*, vol. 2, pp. 1031–1034, IEEE, 2010.

[57] Z. Wang and M. Hassan, "How much of dsrc is available for non-safety use?," in *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, pp. 23–29, 2008.

[58] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[59] J. A. Hatem, A. R. Dhaini, and S. Elbassuoni, "Deep learning-based dynamic bandwidth allocation for future optical access networks," *IEEE Access*, vol. 7, pp. 97307–97318, 2019.

[60] N. Dutta, H. K. D. Sarma, and Z. Polkowski, "Cluster based routing in cognitive radio adhoc networks: Reconnoitering sinr and ett impact on clustering," *Computer Communications*, vol. 115, pp. 10–20, 2018.

[61] T. R. Andel and A. Yasinsac, "On the credibility of manet simulations," *Computer*, vol. 39, no. 7, pp. 48–54, 2006.

[62] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, and N. Sulieman, "Coacs: A cooperative and adaptive caching system for manets," *IEEE Transactions on Mobile Computing*, vol. 7, no. 8, pp. 961–977, 2008.

[63] M. J. Khabbaz and H. A. Artail, "Deadline-constrained connection request scheduling in mobile relay-assisted lte networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6937–6950, 2019.

[64] M. Khabbaz and H. Artail, "Deadline-aware ue service scheduling in mobile-relay-augmented cellular networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2019.