National College of Ireland

Developing a QoS and Spatially aware scalable fog system with adaptive cache

MSc Research Project
Cloud Computing

Oluwashola Akanni
Student ID: x22176098

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Oluwashola Akanni
Student ID:	x22176098
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	14/12/2023
Project Title:	Developing a QoS and Spatially aware scalable fog system with adaptive cache
Word Count:	5949
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Oluwashola Akanni
Date:	31st January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Developing a QoS and Spatially aware scalable fog system with adaptive cache

Oluwashola Akanni
x22176098

Abstract

With the expansion in the number of mobile systems integrated into IoT and fog networks, fog computing now includes scaling out and optimizing communication for mobile fog systems effectively. This research focuses on developing a QoS (Quality of Service) and spatially aware scalable fog system with an adaptive cache to enhance the efficiency and reliability of the system. In order to maintain a system that knows the approximate location of its devices even when they are constantly moving around. This is to optimize resource allocation, reduce latency, and improve the scheduling for these systems. Making use of existing ideas in fog computing, QoS provisioning, spatial awareness, and some caching techniques, a flat distributed network was developed which is also able to switch to a tiered network when offloading becomes required. The scheduler uses an adaptive cache to keep details of each fog Device in the configuration, it also is able to treat VMs on each fog device as independent machines while retaining QoS-aware properties such as Bandwidth and latency. Key performance metrics which were measured, analysed and compared against other possible more traditional fog systems without spatial awareness and adaptive caching were the energy and bandwidth usage and most importantly the savings in latency to show the new system's efficiency. The application domain of choice is the autonomous car Industry, where latency-dependent decision-making could be crucial in averting disaster. The final results were able to show that keeping the system spatially aware was able to improve the QoS performance of the fog setup.

Keywords— Fog computing, latency, spatial awareness, Quality of Service, IoT.

1 Introduction

Since more modern fog systems are ideally designed to grow and scale continually, while scaling it is important to keep the fog system Quality of Service (QoS) aware as well as avoid bottlenecks in communication. Several methods and technologies have attempted to solve these problems. With the recent increase in the number and types of devices which can be considered a part of a fog system, finding a solution that interconnects these devices together without making a sacrifice of resources means a larger fog system can be treated as a cluster of clusters. For cloud technology, every advancement in fog and edge computing is important to making cloud technology more usable for IoT and other types of networked remote systems and Wireless System networks (WSN). While clustering devices, it is important to take note of the possibility of using a cache to improve the performance and what kind of data is best targeted for cache use since even though

the distributed cache presents more capacity, it is still limited. While a lot of initial research work has focused on network, computation and energy management, caching is just as tricky to deal with in a fog system with several nodes and affects the overall topology. With mobile nodes, the Caching system in place must both be able to react to the spatial positioning of the node and reorganize itself as well as keep track of the QoS required by its parts (nodes and edges). In the course of this research, we aim to strike this balance. To see how a spatially organized system with an adaptive cache is able to improve current performance and resource management for mobile fog systems. The possible areas of application for IoTs today are consistently expanding posing new challenges and a constant requirement for improving the architecture. Gubbi et al. (2013)

Section 2 discusses related work done in the field and discusses concepts such as IoT, WSN, cloud, edge and fog computing, broker and mobility management. Section 4 presents the design, algorithms and architecture of the distributed algorithms and resource management that enable the self-organization of the compute clusters of the fog computing system. mechanism maintains a history of previous transmissions, which can be leveraged based on various strategies in order to route the data in an efficient manner. We also discuss such strategies which can be based on predictive methods. Section 5 implements the algorithms in Java, building out on iFogSim to simulate the results of geographically mobile nodes, and we discuss the differences between the approach and alternative methods. Section 6 discusses the results and Section 7 the conclusions

2 Related Work

Recent advances in the fog computing field are due to the progress in research made to optimize the performance of the IoT, WSNs, Personal Digital assistants (PDA) and other components of the fog system. A lot of research work right from work done in monolithic hardware computing, distributed computing and scalable systems has also found relevance in fog computing, especially in the search to create a system that seamlessly supports heterogeneous networks. While optimizing heterogeneous networks is beyond the scope of this research, work done in the course of this research would take cognisance of its importance as well.

Several researches have been individually conducted to study how to optimize Fog systems using techniques relating to the topics below but I have here mentioned especially only those particularly of interest to this discussion. This work builds on these contributions and because of the wide range of cloud computing ideas, it would be inefficient to cover every idea but stick to those closely considered and/or adopted. Each factor that ties into the research has been examined and the literature on them is presented here.

2.1 Spatial awareness

In this research, nodes are clustered based on their spatial data, Yin et al. (2021) provides an efficient approach to doing this. This is very much unlike older clustering algorithms like the one proposed in Ding et al. (2005). Where the Clustering is based on Energy awareness or other resource optimizations. Spatial data however is a constantly changing information for mobile systems which will leave a big task of self-organization for the systems. Clustering of nodes allows for much more efficient organisation and can greatly improve task scheduling Shaik and Baskiyar (2018). This would also help in determining

the direction a node is moving but that is beyond the scope of this research. Here we would build routes primarily based on proximity to edges and then nodes. This is to make sure that a moving node can maintain a fairly predictable set of neighbours.

Wireless device identification (Vaidya et al. (2020) presents a model for this) can then be used to track closely its spatial information. Robinson et al. (2021) Clearly explains how changes in the spatial data of a device can be used to recalculate its relative position within the fog system especially also predict latency.
This spatially aware routing would be more useful in mobile nodes and will be the target system. This would attempt to route the workload first to the nearest IoT then the nearest edge and then, propagate upwards until a node with enough resources is found. This also makes it easier to track the resources in each node to predict the nearest suitable node and update the route tables accordingly. This contribution of this research could be important in future topology designs for mobile IoT systems. It is predicted that this feature will result in lower latencies.

2.2 QoS Awareness

QoS provisioning is a central theme of fog computing and also directly can affect task scheduling and caching. It is important during this research to consider adopting a QoS-aware deployment model such as proposed in Brogi and Forti (2017) for applications that can be aware of the QoS requirements, Skarlat et al. (2017) also makes a case distribution or placement within nodes using fog colonies (which we will compare to fog clusters for this research purpose) which are able to offload tasks to fog cells. This approach was able to develop a model more reactive to real-time system changes within the fog system. Huang et al. (2020) approaches QoS from a perspective of resource management and offloading tasks which is based on a priority of tasks set by the QoS. These are all considerations and this research seeks to find the best way to manage the QoS requirements. In a widespread mobile fog system, several devices and applications are involved both requiring and supporting different QoS levels which is based on the particular protocol in use in that system. The QoS of interest include bandwidth, latency and resource utilization.

2.3 Caching in fog systems

Caching isn't a new idea in improving computing performance and its use dates as far back as 1976. Data locality which is key to preprocessing data Lai and Leu (2017), **?** can be used to improve fog systems which gives the idea that local caching can be used to in fact inform task scheduling and this affects optimization of other resources. Various caching techniques Naeem et al. (2022) are prominent in fog systems Application-based caching Almobaideen and Malkawi (2020), Energy Aware caching, Steiner Tree Based and a host of other optionsSu et al. (2015).
The idea to group fog systems while caching as shown in Shooshtarian et al. (2019) shows that remote sensor systems can, in fact, be treated as clusters and cached semi-locally to improve performance further while Gharib (2023) proves the security of this clustered system which is beyond the scope of this research but important to know that this clustering technique rather improves the security of the system. Finally, this research will culminate in developing an adaptive cache that is able to keep up with the highly dynamic workload of a highly mobile fog system.

2.4 Hierarchical Organisation

Hierarchical organization in fog computing is used to structure the fog devices into a hierarchical architecture to efficiently manage resources and data processing. This is used to optimize task offloading and data distribution. Shaik and Baskiyar (2018) outlines the basis for using hierarchical organization based on spatial awareness to optimize data processing and reduce communication overhead. We can thus develop an organisational structure on the assumption spatial proximity translates to network proximity in terms of latency. This is expected to reduce both the overall latency as well as complexity.
It is expected that if the communication protocol and technology are known, the latency is predictable and we can thus create a self-organizing system that will be included in the scope of research. As in fig 1 Truong et al. (2015) A structure like this allows for devices to communicate with each other as well as the edge which can then take advantage to communicate with the rest of the network. This approach can be more efficient in certain scenarios as the research explores.
Compared to a flat network, A hierarchical network can be more adaptive and can react to dynamic conditions to make a decision on the best routing path. The route table may be stored as a sort of distributed data or locally.

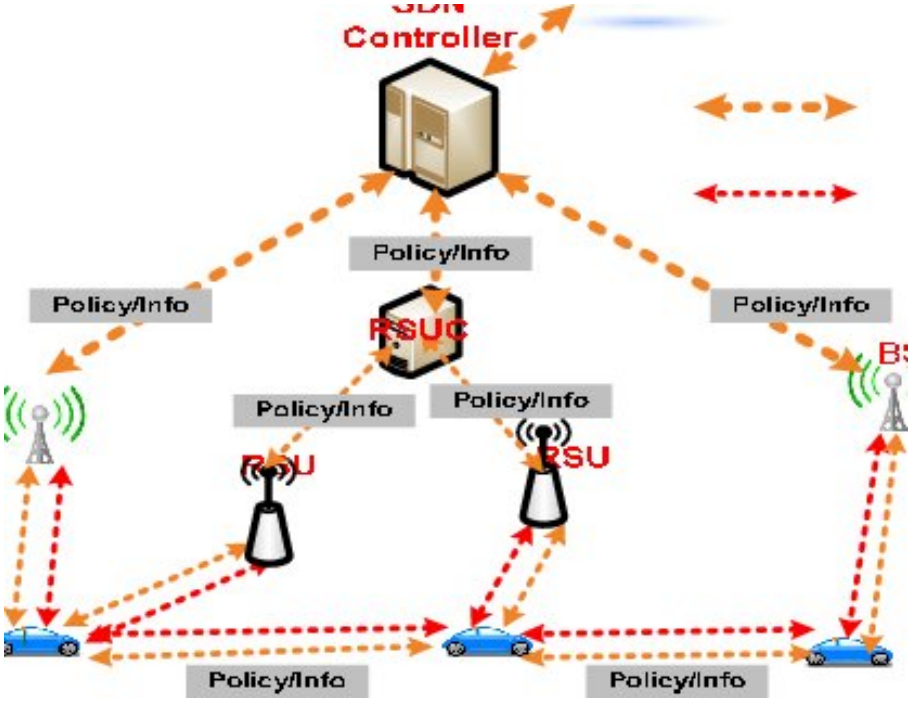

Figure 1: A basic fog system with its interactionsTruong et al. (2015)

2.5 Failure Management

Since resource-constrained devices are much more likely to fail. Research to explore fault detection, and recovery mechanisms to ensure system resilience. which QoS management to an extent will handle certain failures, it is important to have a system in place that can also keep up with failures especially when a distributed cache system is involved. Madsen et al. (2013) considers the possible failures and the proposal is to keep as much

as possible handled within the cluster level. When there is a device failure, the fog system loses resources and potentially also loses information. Reliability is a core factor in the design of fog systems as the failure tolerance is directly intertwined with both the architecture and the respective QoS.

2.6 Scalability

Fog systems face scalability challenges as the range of devices, QoS requirements, protocols and most importantly the growing number of devices. Resource allocation and being able to manipulate tasks as fog networks grow in size may become bottlenecks. the focus is to present a distributed systems approach to this problem. In terms of scalability, clustering as proposed in Shooshtarian et al. (2019) and task offloading as presented in Huang et al. (2020) are great ways to handle the distributed system problem that posed by the fog system in terms of scalability.

3 Methodology

3.1 Problem Scope:

The need for a QoS and spatially aware scalable fog system with an adaptive cache exists due to the recent demands in the field of fog computing. One of such applications is in the context of Autonomous Cars as IoT devices. This research is centred around showing a fog system that allows mobile fog devices to offload workloads directly to each other. The devices are clustered by distance and can offload to other devices within their clusters. The possibility for implementation of the The Wireless Access in Vehicular Environments(WAVE) technology for driverless cars is the idea behind modelling this project around the autonomous vehicle system. WAVE technology is extended past just communication between vehicles to the offloading of workloads between vehicles. QoS metrics of interest here are latency, workload and Bandwidth. The spatial awareness allows the IoT device to move and be added to or removed from clusters, scalability is handled by using a microservice approach, and a shared cache is used to store device information.

3.2 Literature Review:

A review of existing literature on fog computing, QoS management, spatial awareness, scalability, and adaptive caching has been done 2. A major one of the current limits in fog systems and especially autonomous vehicles is the latency due to computation being offloaded to the cloud. However, if vehicle-to-vehicle communication is encouraged and workload is offloaded to nearby vehicles with free resources, this latency can make for faster decision-making of the vehicles.

3.3 Conceptual Framework:

The following parts Clearly define the roles of Autonomous Cars as IoT devices within the fog computing architecture. Specify how QoS will be measured and maintained, taking into consideration factors such as latency, reliability, and bandwidth.

3.4 System Architecture Design:

the architecture Design of this fog system is shown in the design section, detailing the components and their interactions. The Autonomous Cars are in the 4th tier/level of the fog system, they are partially responsible for data processing, communication, and decision-making. For small workloads, all processing is handled within the cluster of the Cars. The offloading is dynamic and first, the system seeks to explore offloading the cluster if this is not available it then offloads to the higher-tier devices.

3.5 Modeling and Simulation:

iFogSim2 is utilized as the simulation tool for the fog computing system. Autonomous Cars are modelled as IoT devices in iFogSim2, simulating their mobility patterns, computational capabilities, and communication characteristics. the QoS metrics and all related algorithms are Implemented within the simulation environment.

Configure the adaptive caching mechanism to reflect real-world scenarios and dynamic changes in the fog environment.

3.6 Parameter Tuning and Validation:

The simulation parameters are Calibrated to closely represent the realistic conditions and characteristics of Autonomous Cars in various scenarios. The results show that there is a slightly lower latency, which could allow for a faster reaction time for these cars. Comparing this to a cloud-centric model shows how much more reaction time the cars have due to round-trip latency.

3.7 Performance Evaluation:

The QoS metrics of note here include the latency and BW. The results of the research have shown a lower latency in clustered distributed architecture for geospatial mobility, scalability, and the adaptability of the caching mechanism under varying workloads and spatial demands.

3.8 Analysis and Interpretation:

An analysis is presented in Section 6 of the simulation results and concludes regarding the effectiveness of the proposed fog computing system. The section also Interprets findings in the context of QoS improvement (in latency), and spatial awareness tracking, and discusses scalability. As well as identify any limitations or challenges encountered during the simulation and propose areas for future research.

3.9 Ethical Considerations:

In the last section, privacy and security concerns are considered associated with Autonomous Cars and the data they generate.

The outlined methodology integrates conceptualization, system design, simulation using iFogSim2, and rigorous evaluation to develop a QoS and spatially aware scalable

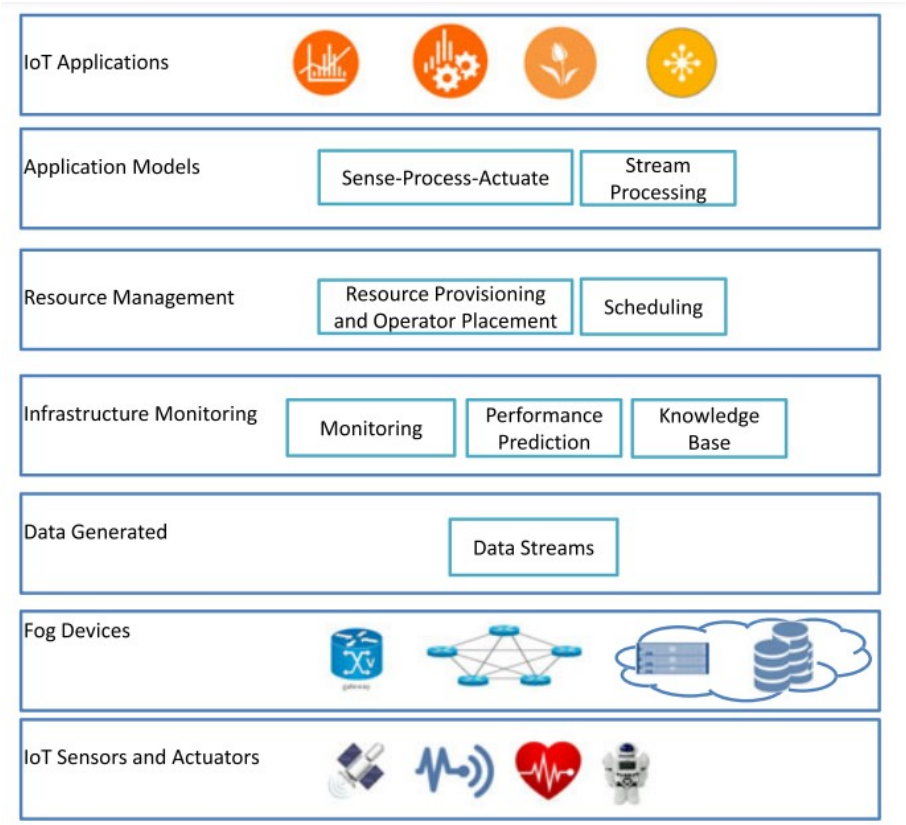

Figure 2: Architecture of Tuple Flow in Fog computing Gupta et al. (2017)

fog system with an adaptive cache, tailored to the unique characteristics of Autonomous Cars as IoT devices.

This project is implemented via simulation using https://github.com/Cloudslab/iFogSim. The base code is adjusted to take the RAM requirements of the workload and target node into consideration. The simulation code also implements the GPS and Gyroscope sensors in order to more accurately model the extra workload contributed by these sensors in reality.

All data used in this project has been generated by the simulation code which is publicly available and publicly licensed and hence ethically sourced. The data generated in different configurations are compared against each other and considered. In order to generate this data, iFogSIm is set to different modes.

4 Proposed Design

The infrastructure for a fog setup usually includes IoT devices(sensors/actuators), some layers of Fog computing nodes (usually the things, the edge and the ISP), and the Cloud (DataCenter).

4.1 QoS

The most important measures of Qos for this work are expected latency and bandwidth, future work may include working on redundancy for the design to handle loss and jitters through a shared cache and status communication with the parent node. However, this

is beyond the scope of this paper. This research seeks to exploit the fact that many IoT devices (This will include mobile phones, autonomous vehicles, and other things that are in close proximity to each other) usually have spare resources at idle. The idea of this work is to bring computation as near to the end device as possible.

Now some assumptions are made in the course of this paper, these include:

1. All fog nodes have signed up legally to offer their idle computing for use up to a certain limit. Say;
   If $computeinuse \geq 50\% * total; computeoffered = 0$
   If $computeinuse \leq 50\% * total; computeoffered = 75\% * (total - computeinuse)$

2. It is assumed that all Fog devices (including things and nodes) have a known location or have geospatial location technologies and this includes modelling a GPS position for all devices.

3. It is assumed all connections are of the same type. i.e. BLE, Wi-Fi, etc. and thus have the same expected latency as a function of distance and are near enough to reach a common endpoint with relatively equal latency as well. This assumption also extends that devices are equally capable of receiving and processing information

4. It is assumed that Fog and Cloud nodes can reach the sensors and actuators of all their neighbour nodes through some middleware APIs that abstract the conversion of tuples to information, and handle the identification of devices, security and other related information.

5. Higher levels in the architecture mean higher compute capacity and the Cloud has a virtually infinite compute capability. It is modelled in the test case using really high numbers compared to the tuple size markings.

4.2 Applications and cloudlets

In the modern era, the use of microservices has been on the continual rise. For the possibility of offloading tasks to similar capacity devices, a divide-and-conquer algorithm using distributed microservices is the best route. If every app can split up its routine tasks and instructions into really small bits interpretable from the APIs the borrowed computing infrastructure can be modelled a distributed architecture and must keep to QoS measures.

The modelling of applications comprises QoS profiles for the interactions between components to express the desired operational systemic qualities, together with hardware, software, and Things requirements for a component to work properly.4 To formalize the notion of compatibility of a software component with a node of a Fog infrastructure, be it a Fog node or a Cloud data centre, Fog nodes must offer the needed software and non-consumable hardware capabilities,

4.3 Cache

For each Cluster, the design includes the use of a distributed cache across the devices. This Cache would contain information on all the other nodes as well as the tasks it is

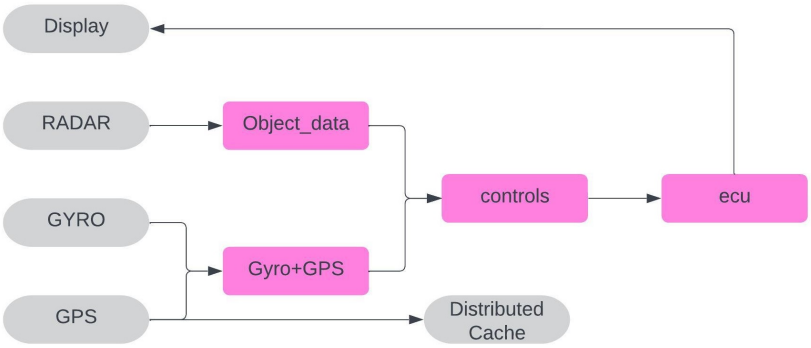

Figure 3: Tuple flow in a sense-process-actuate model

running and the originating (or computing) nodes. This makes it easy in the case of failure to keep the edge informed of what tasks need to be waited for or transferred. This is simply modelled in the simulator using hashmaps. The routes will be determined and updated more dynamically as well based on the spatial data and cluster membership. The parallel Djikistra shortest path routing algorithm has been used in this work to take advantage of the distributed Cache. The fork-join method is also used to simulate what in later work would be the implementation of parallel computing across devices. Using Apache Zookeeper, a partial replication scheme that deploys each component on a subset of surrogates will be followed. A Mobility-Aware Coded Probabilistic Caching Scheme proposed in Liu et al. (2017) and improved by Tang et al. (2019) which allows for Linearly Programming Model for Cache Management for Cloud Radio Access Network (CRAN) and Multi-Access Edge Computing(MEC) using a Zipf Distribution is to be implemented in future work exploiting optimizing the cache in this system proposed.

$$P_{\mathcal{F}}(f) = \frac{\Omega}{f^{\mathrm{A}}} \tag{3}$$

$$\Omega = \left(\sum_{i=1}^{F} \frac{1}{i^{\alpha}} \right)^{-1} \tag{4}$$

4.4 Organisation Topology

It is important for the algorithms to be able to handle disruptions to the organisation. In the proposed system, compute nodes will constantly leave and join the network. each time a new compute node joins the entire system must recalculate its proximity to other nodes. This calculation can be used to estimate its latency to other nodes and decide if it should look to offload to a device higher in the hierarchy or parallel on the same flat topology. The intention is to come up with a more flexible overall algorithm that can theoretically predict the latency measurements and the resources available nearby as discussed under the QoS metrics. In order words the design includes a hybrid of a flat and hierarchical architecture

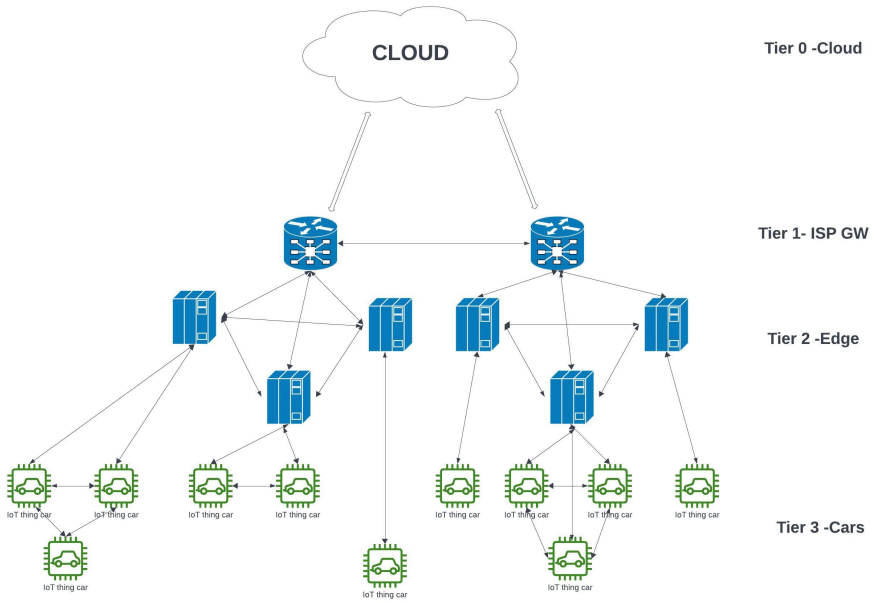

Figure 4: Topology diagram showing Cars in clusters

4.5 Mobility

While the model already includes and models the GPS and Gyroscope sensors input workloads for getting positional information for the model, the mobility and positional values from these sensors are gotten using the RandomMobilityGenerator class to create mobility measures such as car direction, speed, stopping time in each position, and cars' sojourn time Mahmud et al. (2021) with the option to predefine the limits of range of movement from each Edge/Fog node. The random_walk model has been used to generate values for cars which are moving constantly and has been configured to generate different mobility data-sets for each car. Since GPS coordinates are used to model the location of Things, Fog nodes and Cloud data centres, the distance between points is obtained using the Haversine Formula.

4.6 Application design

For multiple compute nodes to be optimally usable, a distributed application architecture is important. It is also cogent to define what processes can run in parallel or serial to each other. Parallel micro-services/processes are independently deployable but may Communicate with other processes. This emphasizes the advantage of future work on a shared cache system. The final tuple could then activate an actuator, trigger another process or simply be saved as a file. The application flow design is represented by a Directed Acyclic Task Graph (DAG), which fits in well with the microservices architecture mentioned and is expected to give more efficient usage of Bandwidth. **?**

It is necessary to introduce the concept of the work done in this project. This design consists of several parts, some of which aren't novel to this research and are simply emphasised to show the choice. The parts of this design include:

- The Application Niche
- The Task Migration

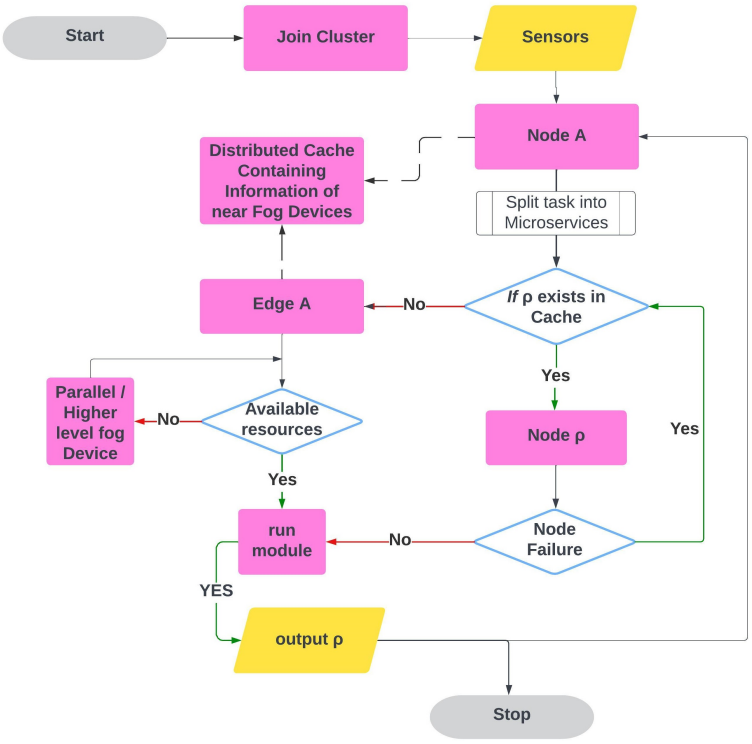

Figure 5: Flow Chart of Task offloading algorithm

- The Topology

In fig4 The Topology of the idea as it relates to the niche of autonomous vehicles with sensors and the feedback to the car screen and the car's ECU is shown. In fig5 The process flow algorithm of the system and the relevance of the cache to the overall architecture are also depicted.

When a sensor detects new information, it sends this to the car ECU which is the fog device. The car ECU creates a couple of tasks and breaks them into task modules, The ECU pings the edge server and is able to get an update of the nearest fog node able to process these tasks. the edge server is able to reply to both the fog node and the other fog node with a task ID for the particular operation. The fog device then forwards its task to the nearest node able to process or the edge device if none exists. The edge devices also periodically send performance reports or metrics to the cloud at certain thresholds in order to keep the cloud prepared for a potential task migration to the cloud if necessary. This information is able to prewarm the resources by starting the necessary micro-services resources like containers before they are needed.

The edge server continues to monitor the synchronisation between both nodes for QoS metrics which are constantly sent to the edge. Where one node cannot reach the other, the edge is immediately notified and the task is migrated to the edge server.

The final stage is the final feedback from processed data to either the edge or synchronised node and finally to the application on the initiating node and/or actuator.

The first Algorithm helps to understand the mobility management for task scheduling. This is the shortest path algorithm that keeps track of the nearest device to each device in order to maintain the most optimal communication option.

The next algorithm from Mahmud et al. (2021) deals with implementing the clustering

for the fog system.

Algorithm 1 Parallel Dijkstra's Algorithm for Fog System using ForkJoin

```
1: procedure ParallelDijkstraForFog(List of FogDevices)
2:     Start Parallel Dijkstra for Fog System
3:
4:     Initialize Routing Matrix, Distance Matrix
5:
6:     Create DijkstraTask for Range of FogDevices
7:
8:     if range exceeds minimum distance  then
9:         Method of DijkstraTask (Left)
10:        Method of DijkstraTask (Right)
11:        indirectDist()
12:        directlyConnectedDist()
13:    end if
14:
15:    Recursive Dijkstra on Left and Right Tasks
16:    Combine Outputs
17:    Update Routing Matrix[][] and Distance Matrix[][]
18:    generateRoutingTable(List¡FogDevice¿ fogDevices)
19:    End
20: end procedure
```

5 Implementation

I have chosen to implement a case scenario for my proposition building out on the iFogSim2 which is the iFogSim updated to be compatible with mobility. Mahmud et al. (2021) The fog computing system is presented in an organisation that includes geo-mobile devices forming distributed compute nodes, Edge devices with reasonable compute power, Gateway Devices with limited compute and the Cloud. We model the latency of the system to be directly dependent on the distance between devices. Hence clustering the devices by a distance which translated to a latency range in practice.

The implementation follows the design previously designed and seeks to do so with autonomous vehicles.

5.1 Autonomous vehicles

Autonomous vehicles use multiple sensors to collect environmental and personal vehicular data that can be analysed to predict future traffic-related occurences like a car speeding up, an accident within a short distance requiring actuators like emergency brakes etc. The cars' ECU is modelled in this work as a module tied to each car permanently as it controls actuators while the rest of the tasks are modelled as modules that can be offloaded to other fog devices. The application domain here is useful for the Autonomous Vehicle Industry Philip et al. (2019). While the proposed IEEE wave technology, which is composed of the IEEE 802.11pJiang and Delgrossi (2008a) and IEEE 1609.x protocolsJiang and Delgrossi

Algorithm 2 Dynamic Distributed Clustering(DDC)

Ensure: procedure $MANAGECLUSTERING(f, t, loc, lf)$

$lHI \leftarrow loc.locationInfo()$
$\rho \leftarrow f.getParent()$
$\eta \leftarrow \rho.getChildren()$
$\delta \leftarrow f.getRange()$
$\sigma \leftarrow f.getLatencyThresh()$
$list_f^{cm} \leftarrow null$
$mapCMToLatency \leftarrow$
$f_x \leftarrow lHI.get(f).lat$
$f_y \leftarrow lHI.get(f).long$
for do $f' := \eta$
    $f'_x \leftarrow lHI.get(f').lat$
    $f'_y \leftarrow lHI.get(f').long$
    $flag \leftarrow calculateInRange(f_x, f_y, f'_x, f'_y, \delta)$
    if f **then**lag
        $list_f^{cm}.add(f')$
        $latency \leftarrow checkLatency(f, f')$
        $mapCMToLatency.get(f') \leftarrow latency$
    end if
end for
if l **then**f then
    $temp \leftarrow list_f^{cm}$
    for do $f' := list_f^{cm}$
        if then $mapCMToLatency.get(f') > \sigma then$
            $temp.remove(f')$
            $mapCMToLatency.remove(f')$
        end if
        $list_f^{cm} \leftarrow temp$
    end for
end if
return $list_f^{cm}, mapCMToLatency$

(2008b) has not been fully implemented as of yet, this research models the expected improvement in the latency which while little could be significant in the domain. The data collected many times also need complex machine learning algorithms and these processes usually sit on the cloud and are not latency sensitive and so are left out of the scope of implementation here. The sensors on these cars modelled include multiple RADARs, as well as GPS and Gyroscope sensors to deliver real-time vehicular data that can be used to predict traffic situations and Actuators that can react. Information from other cars would also be taken into consideration and are assumed to be already integrated by the model. We design an application using microservices and clustering by implementing the MicroservicesMobilityClusteringController class of iFogSim. We have also used a DAG-based model to control the flow of data.

Fig 3 presents the architecture of the application as microservices.

5.1.1 geospatial microservices

This microservice and its components are mainly concerned with the positional details of the Car. It takes inputs from the GPS and gyroscope to both continuously update its current location as well as predict all its likeliest possible future positions. This microservice is very useful both directly to the fog system for clustering and offloading decisions but even more so to the autonomous vehicle ecosystem in preventing accidents, predicting traffic conditions, and self-correction for other vehicles impacted by its movement.

5.1.2 object detection

This microservice is responsible for taking input from several radar sensors which are sending updates at a high frequency and converting this radar information into a sort of model of its environment. This helps the car extract features around it and predict trajectories of these features, helping with immediate decision-making and is uploaded to the cloud for further processing which is outside the scope of this work.

5.2 Simulation

We present the metrics used in the simulation as parameters. The assumed values are within the range of normal operation. For computing the cars are assumed to have decent resources available but with a caveat which is that these resources vary in time. We include the simulation for varying resource levels. The rest of the fog system is hierarchical with more resources the tier level increases with the highest tier at tier 0(the cloud).

Table 1: Description of tuple types in the Autonomous car application

Tuple type	**CPU (MIPS)**	**N/W length**
RSENSOR	2000	500
GSENSOR	200	200
PSENSOR	50	100
OBJECTS_REF	4000	2000
COMMAND	2500	500
LIVE_VIEW	1000	500

Table 2: Configuration of fog devices

Configuration	Cloud VM	ISP_GW	Edge_GW	Car
Numbers	1	12	118	50
CPU(mips)	44800	2800	2800	2000
RAM (MB)	40000	4000	4000	6144
Uplink (mbps)	100	10000	10000	10000
Downlink(mbps)	10000	10000	10000	270
Busy power(MJ)	16 * 103	107.339	107.339	87.53
Idle power (MJ)	16 * 83.25	83.4333	83.4333	82.44

Table 3: Fog device configurations

Module ⇓	RAM(GB)	CPU length(MI)	Size(B)
Object$_data$	512	800	1000
gyro+gps	128	400	400
Controls	256	1000	1000
ecu	128	150	500

Table 4: Module computation requirements

In tier 4, The system has the lowest level of actuators and sensors with no processing power, In tier 3 it has the IoT devices which are cars in this research, in tier 2 it has edge gateway devices followed by tier 1 proxy servers and Cloud at tier-0.

An Intel(R) Core(TM) i5-3437U CPU @ 1.90GHz (Peak 2.40 GHz) with an 8GB-RAM configured computer has been used to execute the simulation script and perform the experiments. The numeric values of the simulation parameters have been recorded in this report. The Number of cars used was 50.

The simulation is done in Java using ifogsim2. Adjustment is made to the base code of iFogSim2 to take RAM resource usage into account. It takes an input of fog device characteristics like processing power, RAM size, power utilization and tier level. The outputs produced by this simulation include latency, energy usage and Bandwidth.

6 Evaluation

We implement 3 experiments to compare the performance of the designed system to other possible configurations.

We compare the following system configurations;

- A cloud offload System
- A hierarchical architecture
- a distributed architecture.

For each of these architectures, we also compare their performance in a distributed micro-service architecture vs monolith offloading. These configurations are also compared under vastly varying workloads. For each of these configurations, the tables and charts show the results below:

Table 6: Configuration of fog devices

Results	Cloud VM	ISP_GW	Edge_GW	Car
Network usage (mbps)	100	10000	10000	10000
Downlink(mbps)	10000	10000	10000	270
Power(MJ)	16 * 103	107.339	107.339	87.53

Table 7: Fog device configurations

6.1 Case Study 1/ Cloud-Centric

The Cloud-Centric deployment is done by directly deploying all modules to the cloud. This removes the latency due to decision-making at each layer, however, it requires a complete round-trip latency to and from the cloud for every tuple transmitted.

6.2 Case Study 2/Hierarchical

The Hierarchical Architecture on the other hand introduces a slightly higher latency due to decision-making at that level, but in most cases is able to avoid workload that goes all the way up the hierarchy to the cloud. When the workload is handled at a lower layer, the latency is much reduced giving better average latency. it can also handle a higher workload than 6.3 . . .

6.3 Experiment / Case Study 3

The Clustered peer-to-peer offloading is able to take advantage of nearby devices to run certain micro-services and hence reduce latency. However, the computing capacity at this level is lower and depends on having a well-populated cluster.

. . .

6.4 Discussion

The comparisons of these results are shown in the charts and overall point to the system being able to make latency savings in using clustered micro-services for a geo-spatial mobile system over ordinary edge or cloud offloading.

Some caveats that should be noted regarding the QoS of this system are that communication failures which will require re-transmissions are not taken into context. it is also assumed that the

Adaptive caching here means that the system is able to always have readily available both the most recent information from each device but also retain for longer the most requested nodes. Hence if a node λ_a requests more time-series data about a node λ_i, in order to make a more accurate decision, it would be readily available. The purpose of including adaptive caching in the context of this research was to be able to get approximately the same latency for fetching information from the cache, for all configurations of the System. Hence there are no experiments showing the effects of the cache. This is already abstracted as part of the clustering delay.

In modelling autonomous cars that are able to communicate with each other using the WAVEJiang and Delgrossi (2008b) technology, the cars are allowed to move randomly and the migration of the workload to set to certain QoS thresholds previously discussed.

QoS Metric ⇓	Value
Network usage	4856.406
Latency	62.05999999999987

Table 8: QoS results for designed system

QoS Metric ⇓	Value
Network usage	105692.0
Latency	127.3595

Table 9: QoS results for Edge-Offloading

QoS Metric ⇓	Value
Network usage	1164200.0
Latency	123

Table 10: QoS results for Cloud Centric

The results of different possible systems of migrations are compared and considered if there has been significant enough improvement for different levels of workload.

The designs include:

- Cloud-centric: First we consider a design that directly offloads all workload to the cloud and has no consideration for geolocation. The modules are all pushed to the cloud for execution. The results are communicated down the same pathway to the car. There is no inertia of computation at each device.

- Edge/hierarchical: We connect all fog devices according to a strict hierarchy from Sensor/actuator, Cars, Edges, ISP Proxy and the Cloud Servers. The system attempts to process each module first at its current level and pushes it to a higher level if resource requirements are not met. There is a slight inertia of computation at each of these levels due to decision-making and VM creation.

- Flat/Hybrid: The resources are clustered based on distance (which is directly related to latency under the same conditions) and modules can also migrate within their clusters. Higher-level Fog nodes only get involved if the cluster at that level cannot handle the task.

The results show that because the Edge system had to Eventually Offload to the Cloud, It had an higher latency than the cloud Centric System. However, the more distributed system in our design showed overall latency due to most compute being handled at the lower level as well as much less usage in the Network.

6.5

Many issues are encountered in the use of iFogSim and

7 Conclusion and Future Work

The research Question "How can a spatially aware adaptive cache enhance the performance and resource utilization in a mobile fog computing environments?" has been answered by the research. It is possible to develop such a system as takes the directional

data for geo-mobile devices such as cars, taking into account their time-specific spatial data and using a distributed adaptive cache to hold the relevant information about the devices. It was also shown that this was implementable using the forkjoin algorithm which suggests it would be equally effective over parrallel systems like zookeeper.

7.1 Contributions

7.1.1 Contribution I

The first contribution is adjusting the iFogSim Simulator to take account of the available RAM space on the fog nodes and not only the CPU ability.

7.1.2 Contribution II

The second contribution is the modelling of the QoS and Spatially aware scalable fog system with adaptive cache using the autonomous car and taking into account the positional sensor inputs as workloads which in previous simulations had been completely ignored.

7.2 limitations

A limitation however of this research is that a lot of factors have been partially or fully abstracted for simulation's sake. it would be more accurate to instead individually simulate the effects variations of each of these factors with separate specialized tools tied by a common API, or build out physical real-life models to test the reality of the outcomes. In future work, this would be a preferred approach.

7.3 FutureWork

A key finding of this research has been that geospatial-aware applications when clustered provide an improvement over the traditional tiered fog systems. The implication for this research is that in the future distributed computing for mobile IoT will be more explored and maybe all future mobile devices will share a protocol that makes this feasible between heterogeneous devices at a much larger scale. This approach to fog computing has been shown to be highly efficient by the research findings. Future work also includes simulating the research work over logically seperated nodes and when the WAVE technology is ready, using the technology.

References

Almobaideen, W. A. and Malkawi, O. M. (2020). Application based caching in fog computing to improve quality of service, *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), CORE-C*, pp. 20–27.

Brogi, A. and Forti, S. (2017). Qos-aware deployment of iot applications through the fog, *IEEE Internet of Things Journal, CORE-C* **4**(5): 1185–1192.

Ding, P., Holliday, J. and Celik, A. (2005). Distributed energy-efficient hierarchical clustering for wireless sensor networks, *in* V. K. Prasanna, S. S. Iyengar, P. G. Spirakis and

M. Welsh (eds), *Distributed Computing in Sensor Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 322–339.

Gharib, A. (2023). *Cluster-Based Information-Centric Wireless Sensor Networks Management for Enhanced User Security Satisfaction in the Internet of Things*, PhD thesis, Carleton University.

Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions, *Future Generation Computer Systems* **29**(7): 1645–1660.

Gupta, H., Vahidnbsp;Dastjerdi, A., Ghosh, S. K. and Buyya, R. (2017). Ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, *Software: Practice and Experience* **47**(9): 1275–1296.

Huang, X., Cui, Y., Chen, Q. and Zhang, J. (2020). Joint task offloading and qos-aware resource allocation in fog-enabled internet-of-things networks, *IEEE Internet of Things Journal, CORE-B* **7**(8): 7194–7206.

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

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

Lai, S.-T. and Leu, F.-Y. (2017). Data preprocessing quality management procedure for improving big data applications efficiency and practicality, *Advances on Broad-Band Wireless Computing, Communication and Applications: Proceedings of the 11th International Conference On Broad-Band Wireless Computing, Communication and Applications (BWCCA–2016) November 5–7, 2016, Korea, CORE-B*, Springer, pp. 731–738.

Liu, X., Zhang, J., Zhang, X. and Wang, W. (2017). Mobility-aware coded probabilistic caching scheme for mec-enabled small cell networks, *IEEE Access* **5**: 17824–17833.

Madsen, H., Burtschy, B., Albeanu, G. and Popentiu-Vladicescu, F. (2013). Reliability in the utility computing era: Towards reliable fog computing, *2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP), B*, pp. 43–46.

Mahmud, R., Pallewatta, S., Goudarzi, M. and Buyya, R. (2021). Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments, *Journal of Systems and Software* **190**: 111351.

Naeem, M. A., Zikria, Y. B., Ali, R., Tariq, U., Meng, Y. and Bashir, A. K. (2022). Cache in fog computing design, concepts, contributions, and security issues in machine learning prospective, *Digital Communications and Networks* .
URL: *https://www.sciencedirect.com/science/article/pii/S2352864822001651*

Philip, B. V., Alpcan, T., Jin, J. and Palaniswami, M. (2019). Distributed real-time iot for autonomous vehicles, *IEEE Transactions on Industrial Informatics* **15**(2): 1131–1140.

Robinson, Y. H., Babu, R., Narayanan, K. L., Krishnan, R., Krishnan, R. S. and Paramaivaooan, M. (2021). Enhanced location identification technique for wireless sensor networks, *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 716–720.

Shaik, S. and Baskiyar, S. (2018). Hierarchical and autonomous fog architecture, *Workshop Proceedings of the 47th International Conference on Parallel Processing CORE-A*, ICPP Workshops '18, Association for Computing Machinery, New York, NY, USA. **URL:** *https://doi.org/10.1145/3229710.3229740*

Shooshtarian, L., Lan, D. and Taherkordi, A. (2019). A clustering-based approach to efficient resource allocation in fog computing, *Pervasive Systems, Algorithms and Networks: 16th International Symposium, I-SPAN 2019, Naples, Italy, September 16-20, 2019, Proceedings 16*, Springer, pp. 207–224.

Skarlat, O., Nardelli, M., Schulte, S. and Dustdar, S. (2017). Towards qos-aware fog service placement, *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), CORE-C*, pp. 89–96.

Su, J., Lin, F., Zhou, X. and Lu, X. (2015). Steiner tree-based optimal resource caching scheme in fog computing, *China Communications* **12**(8): 161–168.

Tang, Y., Rajendiran, D. P. and Moh, M. (2019). Cache management for cloud ran and multi-access edge computing with dynamic input, *2019 International Conference on High Performance Computing amp;amp; Simulation (HPCS)* .

Truong, N., Lee, G. M. and Ghamri-Doudane, Y. (2015). Software defined networking-based vehicular adhoc network with fog computing, *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, CORE-B* pp. 1202–1207.

Vaidya, G., Nambi, A., Prabhakar, T., Kumar T, V. and Sudhakara, S. (2020). Iot-id: A novel device-specific identifier based on unique hardware fingerprints, *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), CORE-C*, pp. 189–202.

Yin, F., Yang, L., Ma, J., Zhou, Y., Wang, Y. and Dai, J. (2021). Identifying iot devices based on spatial and temporal features from network traffic, *Security and Communication Networks* **2021**: 1–16.