Implementation and Evaluation of Semantic Clustering Techniques for Fog Nodes

Alhassan Aly
Master in Innovation and Research in Informatics (MIRI)
Universitat Politècnica de Catalunya (UPC)

Supervisor
Prof. Fatos Xhafa

In partial fulfillment of the requirements for the degree of
Computer Networks and Distributed Systems

April, 2020

Acknowledgements

First of all, I would like to thank Prof. Fatos Xhafa of the Barcelona School of Informatics, at the Polytechnic University of Catalonia for advising and providing me with the literature that aided me in my research.

Secondly, I would also like to thank all the professors that have taught and helped me throughout my master studies, providing me with sufficient knowledge to be able to conduct this thesis. Last but not least, I want to express my gratitude to the academic and administrative staff of the FIB, especially Maribel Gutiérrez as she was always available and helped me with any issues that I encountered throughout my studies.

Most importantly, I dedicate this dissertation to all my family and friends who have always supported and motivated me. During the entire master's program, my parents were constantly pushing and believing in me with great encouragement and support.
A special thanks to my friends Pritom Bora, Burak Kilic and my beautiful girlfriend Josephine Blanc for assisting me in many different ways.

Abstract

Growing at an extremely rapid rate, the Internet of Things (IoT) devices are becoming a crucial part of our everyday lives. They are embedded in almost everything we do on a daily basis. From simple sensors, cell phones, wearable devices to smart city technologies, we are becoming heavily dependent on such devices. At this current state, the Cloud paradigm is being flooded by massive amounts of data continuously. The current amounts of data is minimal compared to the amounts that we are about to witness in the near future, mainly because of the 5G deployment expediting and the increase in network intelligence. This increased data could lead to more network congestion and higher latency, due to the physical distance between the devices and the Cloud data centers. Therefore, a need for a new model is paramount, and will be essential in realizing the Internet of Everything (IoE) and the next stage in the digital evolution. Fog computing is one of the promising paradigms, since it extends the Cloud with intelligent computing units, placed closer to where the data is being generated to offload the Cloud. This tackles the issues of latency, mobility and network congestion. In this work we present a conceptual Fog computing ecosystem, where we model the Cloud to Fog (C2F) environment. Then we implement two dynamic clustering techniques of Fog nodes to utilize combined resources, using a semantic description of the Fog nodes' resources and properties of the edge devices. Finally, we optimize the assignment of applications over Fog cluster resources, using Linear programming and a First Fit Heuristic Algorithm. We evaluate our implementation by analyzing the differences between the two clustering techniques.

We perform several experiments to evaluate our implementation, and the results prove that the heuristic optimization of task allocation is much faster and more consistent than the Linear programming solver, as expected. Moreover, the results show that clustering Fog nodes is beneficial in offloading the Cloud and reducing response times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the number of Internet of Things (IoT) devices is exponentially increasing [1], and the data that is being generated from these devices is growing at an accelerating pace. These devices come in a plethora of different flavors, from simple sensors for small tasks, to complex systems made up from various interconnected devices.

The computational demand of different IoT systems can be handled using the current Cloud-based frameworks [2]. However, applications that require real-time responses would be hindered due to the physical distance between the devices and Cloud data centers, which can decrease the Quality of Service (QoS) [3], or, might even render some applications unattainable. In addition, the massive amounts of data flowing back and forth from the data centers, have the potential of congesting the back-bone network [4].

Fog computing is introduced in order to overcome such boundaries, and deal with the increasing number of devices by extending the Cloud to the edge of the network, allowing applications to run in close proximity of where the data is being generated. Professor Salvatore J. Stolfo [5] coined the term Fog computing [6], then it has been picked up by Cisco [7].

The Fog is a layer or several layers, consisting of computing components such as Raspberry Pi devices, routers, switches and Micro-Data Centers (MDC), usually referred to as Fog nodes, which are heterogeneous and Geo-distributed. They are introduced between edge devices and Cloud data centers. They offer infrastructure resources to store, process and manage data closer to where it is being generated [8], along with adding networking functionalities. Therefore reducing application service time due to propagation delay, and the load on the backbone network to avoid bottle-necking and congestion. However, Fog nodes are substantially inferior to Cloud data centers in terms of computational capabilities.

With the 5G broadband mobile network on the horizon, bringing processing closer to the edge of the network is necessary for the technology to reach its full potential, and deliver ultra-high-speeds of data transfer, as well as sub-millisecond latency [9]. This is caused by some 5G communications utilizing high frequency signals in the millimeter-wave frequency band, and with those high frequency signals, several issues such as short range and Line-Of-Sight (LoS) arise [9].

Fog Computing provides solutions for numerous applications, especially for industry 4.0 [10]. An area of application for Fog computing is **Self-driving cars**. The sensors of the intelligent vehicles capture an immense amount of information about the environment and driving conditions, which has to be analyzed and processed within a very short time. In order for the vehicles to respond to current road conditions, traffic conditions and unexpected events and anomalies such as pothole occurrences [11], it is crucial that processing large amounts of data with minimal latency is possible. Edge computing and Fog computing enable data to be processed directly in the vehicle, or processed in a device in the vicinity of the vehicle respectively. At the same time, the network services and central resources are decoupled.

Another, yet closely related scenario would involve Smart Traffic Light Systems (STLS), which is a network of traffic lights that intelligently takes decisions that reduce traffic congestion, minimize noise and fuel consumption, prevent accidents, and give the drivers a better experience by long-term monitoring [12]. Although the STLS is just a component of the larger idea of smart connected vehicles and advanced transportation systems, it is rich enough to drive some key requirements for Fog computing. A traffic light in an STLS should be able to detect vehicles not following traffic rules and inform adjacent traffic lights to notify vehicles or pedestrians that could potentially be affected by those rogue vehicles [13]. However, this accident prevention mechanism causes the traffic light cycles in the affected area to go out of synchronization. Therefore, re-synchronization is required to mitigate this consequence.

Flow control is essential to ensure a smooth movement of traffic without having to make the drivers stop too often. A STLS can collect information about the level of traffic in each lane of the city, and maintain a lane continuously open for a certain period of time, reducing the number of times a vehicle would have to stop at traffic signals. This would reduce congestion, fuel consumption, and noise, since vehicles would not have to accelerate as often [14]. Moreover, by monitoring and analyzing the data gathered from the entire system over a large period of time, the STLS can improve its traffic routing plan continuously. Through long-term analysis on observed vehicle and pedestrian movements, the

STLS would be able to decide the optimal times for which pedestrians should be allowed to cross.

The design requirements of STLS include:

- **Low-latency response**: Accident prevention requires a very low response time to alert the potentially affected entities.
- **Handling a large volume of data**: A STLS has a large number of sensors that generate data at a high rate, poor network architecture can be a victim of bandwidth over-utilization and cause congestion.
- **Heavy processing power**: Planning requires large amounts of data to be processed, and the analysis has to be done on a city level.

Sensors that are deployed on the roads and CCTV cameras installed at intersections, are the main data collection points in a STLS. Sensors can detect crossing vehicles and their speeds. Since any action performed by the STLS is reflected by a change of traffic lights, the traffic lights are considered the actuators of the system. Each intersection could be equipped with a 5G small-cell which connects the devices on that intersection, allowing real-time device-to-device communication among them. These small cells could potentially include a Fog component [15]. The small cell is also connected to the Cloud by a high bandwidth connection through intermediate network devices. These network devices will be used for communicating with other devices in the neighboring intersections, which are also Fog-enabled, meaning that they are points for offloading application logic as well. The device component of STLS are the sensors, CCTV cameras, and traffic lights. The sensors should be able to send updates to the small cells over a 5G network. The CCTV cameras should process the recorded video stream in real-time, to detect events of interest [16]. As for the traffic lights they just change lights according to the decisions taken by the system. Regarding the Fog component, it runs on the small cell in each intersection, as well as on the intermediate network devices connecting the small cells to the Internet. The logic running on these devices handles most of the requirements of a STLS. The Fog component should handle the data sent from the sensors and CCTV cameras and detect possible accidents, then sends a message to the traffic lights on corresponding streets, to change lights accordingly in real time. Finally, the Cloud component is responsible for the long term analysis of the STLS system. Data about the traffic conditions and events generated from the small cells will be uploaded to the Cloud at regular intervals, which reduces the volume of data sent at any given moment to avoid congestion.

Fog computing could also be utilized to semantically enrich data streams with contextual information, as well as complex event processing in IoT applications. One use case would be in the field of e-Health, as an anomaly detection, and classification scenario over an Electrocardiogram (ECG) stream [17]. ECG signals are processed dynamically and classified with modern machine learning algorithms. By applying the algorithms in the Fog layer, the data volume can be reduced, resulting in the reduction of the classification latency, as well as required processing resources.

In this thesis work, we model a conceptual Cloud to Fog (C2F) environment. We implement two dynamic clustering techniques for the Fog nodes, so that they serve applications within a cluster in order to overcome the limitations of Fog nodes, by utilizing their combined resources. We describe the clustering techniques later in the implementation section of Chapter 3. The proposed approach first considers an application consisting of a number of tasks, then identifies and selects suitable Fog nodes to form clusters, and finally sends the tasks to the selected Fog nodes. We use a mapping algorithm built on a unified semantic description, that provides a shared vocabulary for representing application requirements and Fog node properties. To establish the allocation of tasks in a Fog cluster, the proposed solution is optimized using Linear programming (LP) and a First Fit Heuristic Algorithm (FFHA).

1.1 State of the Art

Our proposed approach relies mainly on describing device properties, and application requirements in a unified scheme. It also depends on optimization techniques to manage the task placement and allocation problem. Moreover, simulating the entire environment is crucial for testing and evaluation. Correspondingly, this section presents an overview of existing works in the aforementioned technological backgrounds, with regard to the scope of this project.

1.1.1 Semantic representation

IoT and IoE environments integrate a broad spectrum of interconnected smart objects and networking devices with heterogeneous capabilities. The data flowing within such environments inherently has a huge amount of different compositions coming from different domains, rendering the comprehensibility of the environment rather overwhelming. Therefore, adding a semantic description and representation of the task requirements, properties of the Fog nodes and the connections between them, is highly beneficial in utilizing a given system in an efficient way.

Semantic Web technologies are developed for this purpose, since they facilitate unified data representation, and have been popularly used in the context of heterogeneous IoT environments, with the goal of addressing system interoperability and incorporation related challenges [18]. Using the Semantic Web technology stack to represent data in a uniform and homogeneous manner, with situation awareness across distributed sensing nodes [19]. Semantic Sensor Network Ontology (SSNO) [20] is one of the main products of this initiative, SSNO is a community-directed vocabulary, modeling the sector of physical sensor networks. The Web of Things (WoT) [21] is yet another promising initiative that combines the Semantic Web and the IoT to implement a Web of physical smart things.

1.1.2 Task placement optimization

The decentralized computation within Fog clusters can be significant in improving application response times, and decreasing network congestion, which benefits a number of IoT scenarios such as smart city or smart grid scenarios, where a large volume of data needs to be processed in real-time. Breaking down the applications into tasks, and sending them to a number of Fog nodes to be processed. This raises the issue of optimizing the task placement, considering the variety of the requirements of the tasks and the available resources of each Fog node.

This could be modeled as an Integer Linear Programming (ILP) or a Mixed Linear Programming (MILP) optimization problem to find the optimal solution. One study models the application service placement over Fog resources for IoT as an ILP problem, considering the heterogeneity of applications and resources in terms of QoS attributes. However, in a real life scenario where the number of Fog nodes and edge devices are high, this approach will take a rather long time to find the optimal solution. This shortcoming is then addressed by proposing a Greedy First Fit Heuristic and a genetic algorithm as a problem resolution heuristic [22]. The results show that a faster response time is achieved when using the genetic algorithm, but a better utilization of Fog resources when using the Linear Programming (LP) optimization model.

1.1.3 Environment modeling and simulation

In order to test and evaluate such a complex environment that integrates diverse policies with different configurations, a platform or scheme that enables the quantification of performance of resource management strategies, proves essential. One way to approach this is through modeling and simulation.

iFogSim is a simulation tool written in Java for IoT, Edge and Fog Computing environments [23]. It allows the investigation and comparison of resource management techniques based on certain criteria, such as network congestion, network latency, cost, and energy consumption. Other studies have used iFogSim in their research such as [24] and [25]. YAFS is yet another discrete event simulation tool for Fog and IoT environments, written in Python [26]. YAFS incorporates the simulation of factors such as dynamic link and node failures, user mobility, network congestion, and application popularity, among other aspects. YAFS also accomplishes several design objectives, since it incorporates describing the network topology based on complex network theory, a user customized configuration of policies, a light syntax, and a dynamic invocation of policies during the simulation.

1.2 Research Questions

In this work, we intend to answer the following questions:

- **How to model and simulate the Cloud - Fog environment?**
- **How to allocate applications to Fog nodes within a cluster using semantic description of resources and application requirements?**
- **What are the differences in response times and the number of devices being served in different scenarios?**
- **What are the differences in execution time and the quality of optimization when using LP and heuristic methods?**

1.3 Thesis Structure

The thesis will be structured as follows. Chapter 2 will explain the motivation behind solving the aforementioned research questions, the scope and the objectives of the project. Chapter 3 explains in detail the methodology, architecture and the implementation of the system. Next, in Chapter 4, we present our evaluation scheme of the system, the conducted experiments and assess the results of our proposed implementation. Finally, in Chapter 5, we present our final thoughts, conclusions and future work.

Chapter 2

Motivation, Scope and Objectives

2.1 Motivation

The number of devices connected to the internet is expected to reach over 75 billion devices in 2025 [1]. That is an extremely large number of devices, which will impose a corresponding increase in communications. This makes relying solely on the Cloud paradigm infeasible. Several studies present and explain edge and Fog computing, and how they could be the go-to solution to compensate for such massive amounts of data, and for good reason.

From providing some insights on service placement in Cloud to Fog (C2F) environments in [27], to presenting a conceptual framework that tackles the resource provisioning problem [25], to [28], where a profit-aware application placement policy for integrated Fog-Cloud environments is proposed.

These efforts all converge to serve in realizing the Fog computing paradigm, which promises to reduce service latency and enable critical applications requiring real-time responses, reduce the network congestion and the energy consumption, while also bringing beneficial security aspects [27]. Fog nodes are also known as Micro-data centers, Mini-Clouds or Cloud-lets, see [29] for insights on how Fog nodes may be defined.

The Fog computing principle has not yet matured enough to be successfully adopted for commercial and industrial use. However, the theoretical foundations are mostly well established [25]. Recent researches enabled the grouping of computing resources from multiple edge devices, to handle data-intensive tasks using Big Data clustering middle-ware. The use of these solutions is still being held back by the resource constrictions on the devices, the device variety, and the time-critical, mobile and alternating nature of IoT environments [30]. A dynamic clustering technique for Fog nodes could aid in addressing these issues.

Regarding this thesis work, we implement and evaluate two dynamic clustering policies of Fog nodes, based on a semantic description of Fog resources and application requirements, enabling the processing of more demanding applications.

2.2 Scope

The scope of this thesis includes the modeling and simulation of the C2F environment for testing different scenarios. These scenarios are constructed to compare between Cloud-to-device and Fog-to-device communications in terms of response times, based on a unified semantic description of resources and application requirements. Furthermore, we address the task allocation problem with two optimization techniques.

Issues regarding 5G (line of sight, wave frequency, etc..) are not considered in this thesis. Software compatibility between applications and Fog nodes is key for realizing Fog computing architectures. Furthermore, job handling and scheduling are essential features [30]. Security of the Fog nodes and the communication links is evidently a major concern [28]. However, the aforementioned aspects are out of this project's scope, and would rather be considered as related research or further future work.

2.3 Objectives

The project has several objectives that consist of (1) modeling and simulating the C2F environment; (2) optimizing task allocation; (3) evaluating the proposed clustering techniques;

Regarding the constraints mentioned above, we define the problem statement as: **How to model and simulate the C2F ecosystem, using semantic description of Fog devices and applications, to allocate given tasks to offload some of the processing load from the Cloud?**

To tackle this problem, in the section below we define a set of objectives:

- Modeling and simulating the environment consisting of:
  - Cloud entity
  - Fog nodes
  - Edge devices
- Clustering Fog nodes to serve a given application
  - In-range Clustering

  - Neighbor Clustering

- Optimizing the task allocation in a cluster using:
  - Linear Programming
  - Heuristics
- Evaluating the proposed implementation in terms of:
  - Application response times
  - Bench-marking the optimization techniques
  - Assessing the utility of the clustering techniques

The clustering techniques mentioned above will be explained later in the implementation section of the next chapter. Finally, we will evaluate the results by comparing the information gathered from our tests presented in Chapter 4.

Chapter 3

Methodology, architecture and implementation

3.1 Outline

In this chapter, we present our methodology of our implementation, and we explain the architecture of the system and its components. Then we describe how we modeled the simulation environment.

3.2 Methodology

We approach the problem by viewing the system as a network, consisting of a number of nodes. This network consists of a Cloud node, multiple Fog nodes and edge devices. On one hand, the Cloud node will always have the same coordinates that can be set prior to the simulation. The Fog nodes and edge devices, on the other hand, are going to be generated with random coordinates following a uniform distribution. The number of generated nodes is set before running the simulation.

We first assume that the Fog nodes are connected to the Cloud node by default, regardless of the distances between them. However, concerning the edge devices, they will not have any connections to either the Cloud node, nor to any of the Fog nodes initially. We can then start connecting edge devices to Fog nodes, if they fall within a predefined coverage range set for each of the Fog nodes. We assume the distances from the Cloud to the Fog nodes to be around **5km**, being the estimated distance between central Barcelona (Plaça de Catalunya) and Barcelona Super-computing Center (BSC).

As the scope of this project is not concerned with application deployment and job handling, we assume that each edge device will have only one request, or application, made up of a number of tasks with certain requirements.

We model three scenarios:

1. When a device does not fall in range of any Fog node:
    (a) The application will be directly sent to the Cloud.
2. When a device falls in range of only one Fog node:
    (a) This node has the sufficient resources, then the application is sent as a whole to this node.
    (b) This node does not have the resources required by the device, then the node will start checking if there are any neighboring Fog nodes in its range, that could be incorporated to form a Neighbor Cluster.
    (c) There are no Fog nodes in range to form a cluster, or the potential neighbor clusters could not handle the application, then the request

is forwarded to the Cloud through the initial node that received the request.

3. When a device falls in range of multiple Fog nodes:
   (a) At least one node has the sufficient resources, then the application is sent as a whole to the node which gives the shortest response time.
   (b) None of the nodes can solely handle the request, then they check if more than one node can handle the application together to form an In-range Cluster.
   (c) The potential In-range clusters do not have the needed resources, then the in-range nodes will start checking for neighbor nodes to incorporate to form a new neighbor cluster.
   (d) The potential neighbor clusters can not handle the request, then the request is forwarded to the Cloud through the node with the closest proximity to the device.

The choice of which Fog node is the most suitable to incorporate to form either an In-range cluster, or a Neighbor cluster, in order to serve a request, is based on a semantic description of the nodes' resources and application requirements: such as the memory, the compute power of a node, the number of tasks, task size and the number of instructions of a task. The distribution of tasks of an application among Fog nodes within a cluster, is treated as an optimization problem with the objective of minimizing the number of nodes used, and the response time.

3.3 Architecture

As mentioned before in the introduction, Fog nodes could be composed of one or more layers, however in our implementation we consider just one layer between the Cloud and the edge devices.

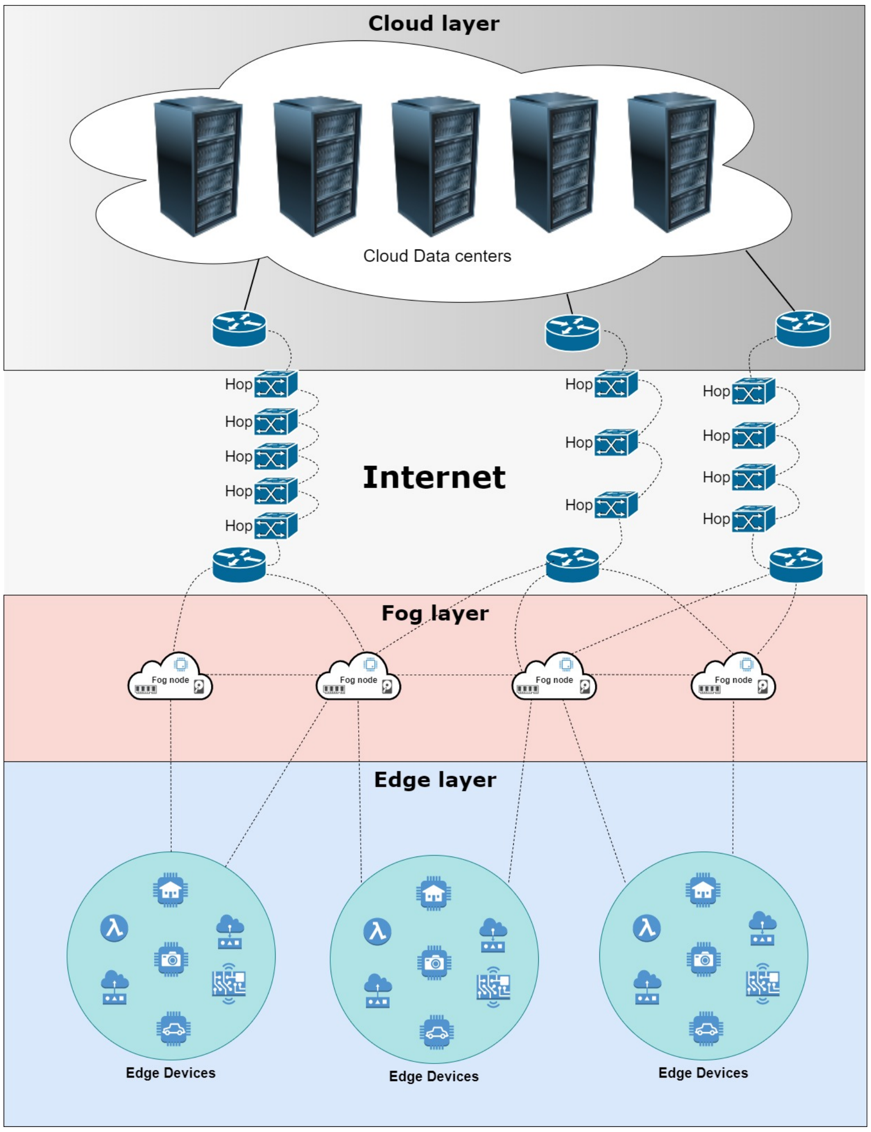


Figure 3.1: Architecture overview

As shown in Figure 3.1 we first have a Cloud layer which contains the Cloud infrastructure, then the Fog layer containing the Fog nodes which are connected to the Cloud through the internet. The connections between the Cloud and the Fog nodes pass through several hops. The physical distance between the Cloud data centers, and the Fog nodes affects data transfer rate, since it increases both latency and potential packet loss. The Fog nodes are directly connected to the edge devices i.e one hop away, which makes the data transfer faster, more stable and reliable.

The Cloud node has abundant memory and processing power, while the Fog nodes are rather limited in comparison. The edge devices in our system will just have one application each that they need to run using certain specifications.

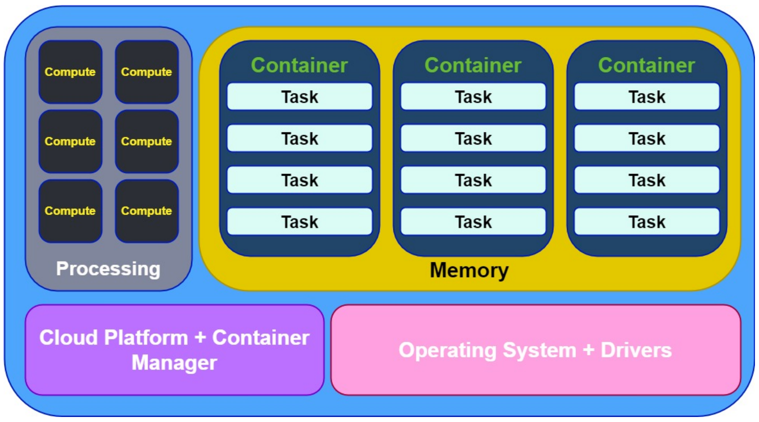


Figure 3.2: Cloud architecture

We describe the Cloud in Figure 3.2 as a system that consists mainly of a processing component, memory component, management and operating systems. The memory incorporates containers as virtualization technology for application delivery and execution.

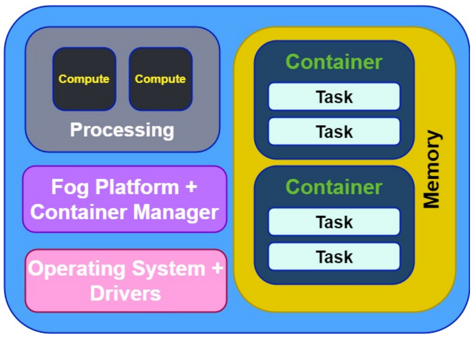


Figure 3.3: Fog node architecture

Similarly, the Fog nodes as shown in Figure 3.3 consist of the same building blocks of the Cloud, but with far more less memory and computational power. Fog nodes may benefit greatly from containers because of their lightweight and resource-friendly structure [31].

3.4 Implementation

We implemented the simulation using Python 3.8.2, which is the latest stable version of Python as the time of writing and development. The main library used is the NetworkX version 2.4, which enabled us to define and manipulate the nodes, and the connections between them. Additionally, the library includes a number of useful functions that create a myriad of options for creating different networks for testing and analyzing the results.

3.4.1 Environment modelling

In this project, we describe the Cloud and the Fog nodes as computing entities that contain CPU and memory. In real life there are several other factors that determine the performance of a machine such as memory bus speed, and type of storage technology just to mention a few.

3.4.1.1 Fog nodes

We describe each node with the following attributes:

- **MIPS**: Million instructions per second
- **RAM**: Memory capacity
- **Range**: Range of node in meters

Realistically, the Cloud has elastic memory and compute power, which are expandable if needed. In our system, the Cloud will have a fixed **5000 MIPS** and **64000 MB** of **RAM**. We consider that the Fog node resources could potentially be similar to Raspberry Pi like devices. Therefore, in our implementation we use the specifications and benchmarks of Raspberry Pi models 4, 3A+, 3B+ and B

[32] for the Fog node attributes. As Fog nodes could potentially be placed within 5G small cells [15], they will inherently have the same coverage range. Thus, the Fog nodes will be generated at random positions and will have varying attributes ranging from **200 MIPS** to **900 MIPS** for compute power, and from **512 MB** to **4096 MB** of **RAM**, and **500 m** to **2500 m** for the **Range**.

3.4.1.2 Edge Devices

Similarly we model the applications that the edge devices are going to request using the following attributes:

- **Tasks**: Number of tasks
- **Tsize**: Size of each task
- **TmINS**: Number of instructions in each task (in millions)

How an application would perform in the real world depends on a lot of different factors such as complexity, inter-dependencies and third-Party components, among many others. However, we follow a more simple path with Edge devices similar to their Fog counterparts, where they are also generated at random positions and assigned random attributes. Each application will have both a common **Tsize** and a common **TmINS**, ranging from **32 MB** to **256 MB**, and from **1 million instructions** to **5 million instructions** respectively. The number of tasks ranges from **1** to **100 Tasks**.

3.4.1.3 Connections

As for the connections between the nodes, we assign them the following attributes:

- **DR**: Data Rate

For the values we used for the data rates, we followed the theoretical performance metrics regarding the 5G broadband cellular network [33]. We apply this only for the connections between Fog nodes and edge devices, for the connections either between Fog nodes and Cloud, or edge devices and Cloud we follow the current values of Ethernet and 4G network [34] [35]. Since different environments will have different data rates, we generate data rates for the connections at random, ranging from **1000 Mb/s** to **10000 Mb/s** for the Fog to device connections, and from **100 Mb/s** to **1000 Mb/s** for the Fog to Cloud and device to Cloud connections.

3.5 System Workflow

As mentioned before, the Cloud node is always generated in a fixed position with fixed attributes, the Fog nodes are generated at random positions with random attributes, and are all connected to the Cloud node. Then, the edge devices are also generated at random positions, with random attributes for their applications. However, the edge devices initially do not have any connections.

3.5.1 Discovery and selection

An edge device has to fall within range of a Fog node in order to have the potential to connect with that node. Furthermore, if a device falls in range of multiple Fog nodes, they will all be added to a candidate list which will be used to find which Fog node is best for serving the application or for creating potential In-range clusters.

3.5.2 Semantic reasoning and clustering

Since we have a unified description of the Fog node resources and the requirements of each application in the edge devices, we can formulate an algorithm to determine how the application is sent, whether through a single Fog node, an In-range cluster, a neighbor cluster, or directly to the Cloud.

The algorithm first adds all Fog nodes in range of a device to a candidate list, then performs a memory check on the nodes in that list to find the nodes that could handle the application exclusively. The memory check verifies if the memory of a Fog node is enough to accommodate the application depending on its size, then if there is one or more nodes that pass the memory test, their corresponding response time is calculated, and the application is sent to the node with the minimum response time.

In the case that all nodes in the candidate list fail the initial memory check, the algorithm performs a combined memory check by using different combinations of the nodes in the list to check if any combination has the required resources. The combinations that pass the check are considered as potential In-range clusters. Then, the application's tasks are divided among the nodes in the cluster that gives the minimum response time, and uses the least number of nodes.

If all the combinations did not pass the second memory check, then the nodes in the candidate list will check for any neighbor nodes in range. The algorithm then performs another combined memory check for all the possible combinations of in range and neighbor nodes. The combinations that pass the check are considered as potential Neighbor clusters. The application's tasks are then divided among the nodes in the cluster that gives the minimum response time, and uses the least number of nodes.

Supposing that all potential neighbor clusters fail the third check, then the application is forwarded to the Cloud through the node in the candidate list that has the fastest link to the device. Moreover, if a device does not fall within a range of any of the generated Fog nodes, its application is directly sent to the Cloud.

We calculate the response time using the following equations:

$$\boldsymbol{Response\ time} = Processing\ time + Network\ Latency$$

Where:

$$\boldsymbol{Processing\ time} = \frac{Number\ of\ Instructions}{Number\ of\ Instructions\ per\ Second}$$

$$\boldsymbol{Network\ Latency} = Propagation\ Delay + Serialization\ Delay$$

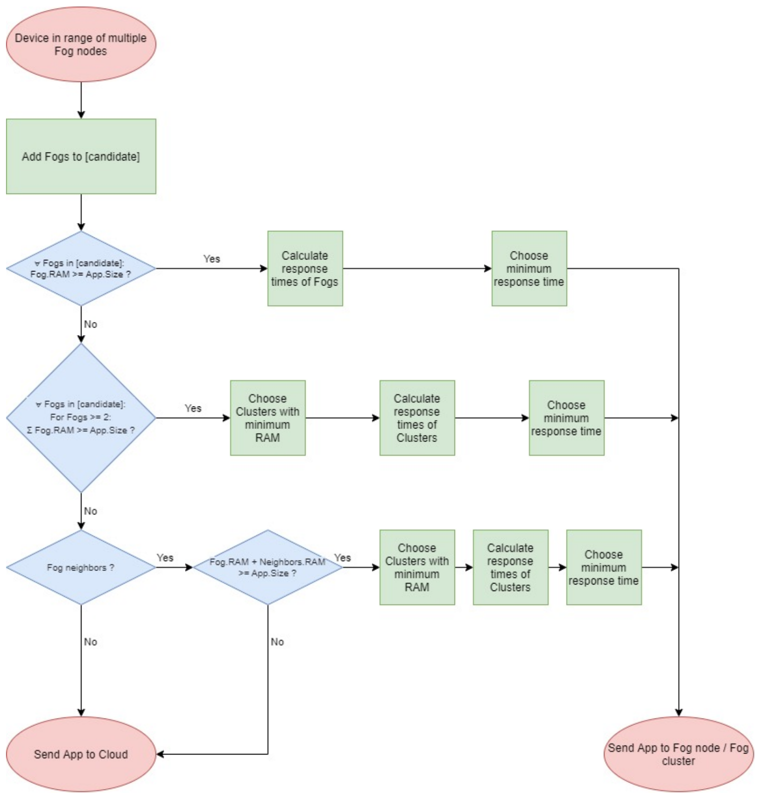
And:

$$\boldsymbol{Propagation\ Delay} = \frac{Distance}{Speed\ of\ Medium}$$

$$\boldsymbol{Serialization\ Delay} = \frac{Package\ Size}{Data\ Rate}$$

Regarding the ***Processing time***, we already have the ***Number of Instructions*** and the ***Number of Instructions per Second***, from the edge device and Fog node attributes respectively. For the ***Propagation Delay***, we already have calculated the ***Distance*** between any given Fog node and any edge device, and we assume that the ***Speed of Medium*** is equal to the speed of light. As for the ***Serialization Delay***, the ***Package Size*** is the number of tasks in a given edge device multiplied by the size of the task, and the ***Data Rate*** is already defined as an attribute to each connection.

Figure 3.4 and Figure 3.5 show the workflow of when a device falls in range of multiple Fog nodes and when a device is in range of just one Fog node respectively.



Device in range of multiple Fog nodes
Add Fogs to [candidate]
∀ Fogs in [candidate]: Fog.RAM >= App.Size ?
Yes
Calculate response times of Fogs
Choose minimum response time
No
∀ Fogs in [candidate]: For Fogs >= 2: Σ Fog.RAM >= App.Size ?
Yes
Choose Clusters with minimum RAM
Calculate response times of Clusters
Choose minimum response time
No
Fog neighbors ?
Yes
Fog.RAM + Neighbors.RAM >= App.Size ?
Yes
Choose Clusters with minimum RAM
Calculate response times of Clusters
Choose minimum response time
No
No
Send App to Cloud
Send App to Fog node / Fog cluster

Figure 3.4: Edge device in range of multiple Fog nodes

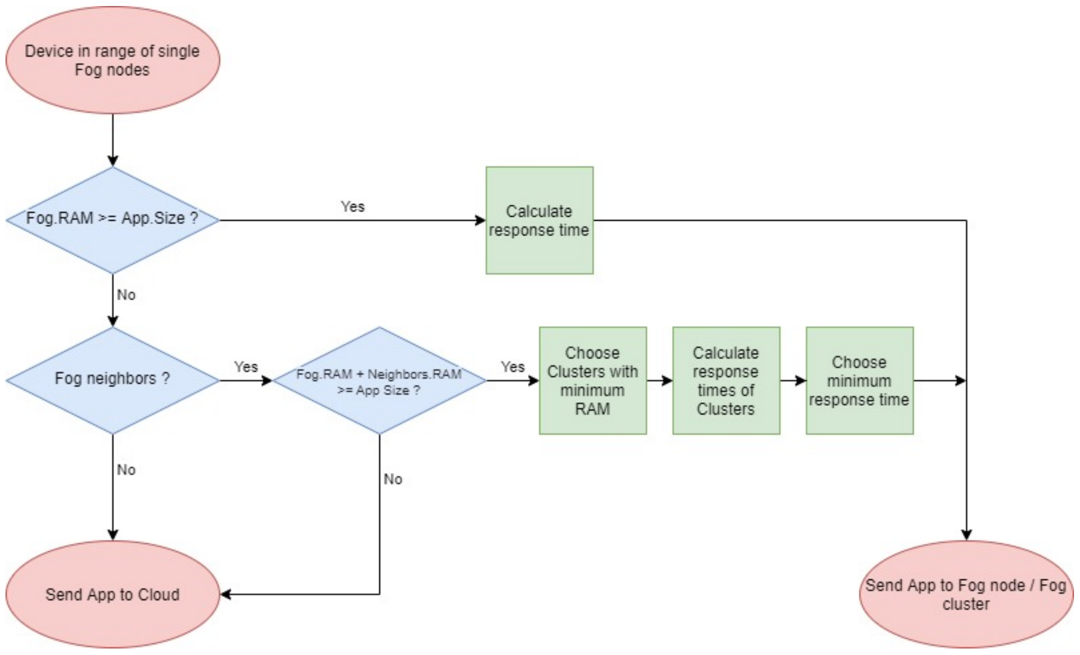

Device in range of single Fog nodes
Fog.RAM >= App.Size ?
Yes
Calculate response time
No
Fog neighbors ?
Yes
Fog.RAM + Neighbors.RAM >= App Size ?
Yes
Choose Clusters with minimum RAM
Calculate response times of Clusters
Choose minimum response time
No
No
Send App to Cloud
Send App to Fog node / Fog cluster

Figure 3.5: Edge device in range of a single Fog node

3.5.3 Task allocation optimization

The responsibility of the solver in our system, is to perform the mapping between the tasks in an application to Fog nodes within a cluster. Its function is to find which Fog node is going to execute which tasks of an application. Thus, mapping could be considered as an optimization problem. The problem could be described as a variant to the bin-packing problem [36], where we have a number of items that we want to place in a number of bins, with a common capacity and minimize the number of used bins. Unlike the traditional problem, we will have different capacities to represent the memory of the Fog nodes. We first apply ILP to get the optimum solution, then we solve the problem using a First Fit Heuristic Algorithm (FFHA) [37] approach to decrease the run-time of the optimization.

The objective is minimizing both the number of Fog nodes used, and the processing time by allocating as many tasks to Fog nodes with higher CPU as possible.

We formalize the problem as follows:

Objective:

$$\min \sum_{f \in F} f_{p_t} \times f\_used$$

Where:

- f: Fog node.
- F: Set of Fog nodes.
- f_{p_t}: processing time of node *f*.
- f_used: equals 1 if node f is used, 0 otherwise.

subject to:

- constraint 1:

$$\forall t \in T:$$

$$\sum_{f \in F} x_{tf} = 1$$

- constraint 2:

$$\forall f \in F:$$

$$\sum_{t \in T} t\_size \times x_{tf} \leq f_{RAM} \times f$$

Where:

- x_{t_f}: equals 1 if task t is placed in Fog node f, 0 otherwise.

- t_size: Size of task t
- f_{ram}: RAM capacity of *f*.

The constraints 1 and 2, make sure that a task can only exist in one Fog node, and that the sum of task sizes cannot exceed the memory capacity of the Fog node they are placed in respectively.

For the ILP method, we use the *PuLP* Python package to solve the optimization problem. *PuLP* can generate Mathematical Programming System (MPS) or Linear Programming (LP) files and call GLPK [38], COIN CLP/CBC [39], CPLEX [40], and GUROBI [41] to solve linear problems. PuLP uses Coin-or Branch and Cut (CBC) as its default solver, which is an open-source mixed integer programming solver written in C++.

For the FFHA, we define our own algorithm to solve our specific problem, where we have a variable number to represent Fog node's **RAM**. The following snippet shows the pseudo-code of the FFHA implementation:

Algorithm 1 First Fit Heuristic Algorithm

```
for $Fogs = 1, 2, \dots$ do
    for $index = 1, 2, \dots, inRangeFogSizes.length$ do
        Assign $taskSize$ in sizes to $sizes[index][1]$
        if $filledFogselectedFogSize \dots$ then
            if $taskSize <= (FogSize - filledFogSize)$ then
                $filledFog + taskSize$
            else
                break
            end if
        else
            return to loop
        end if
    end for
    Return assignment results
end for
```

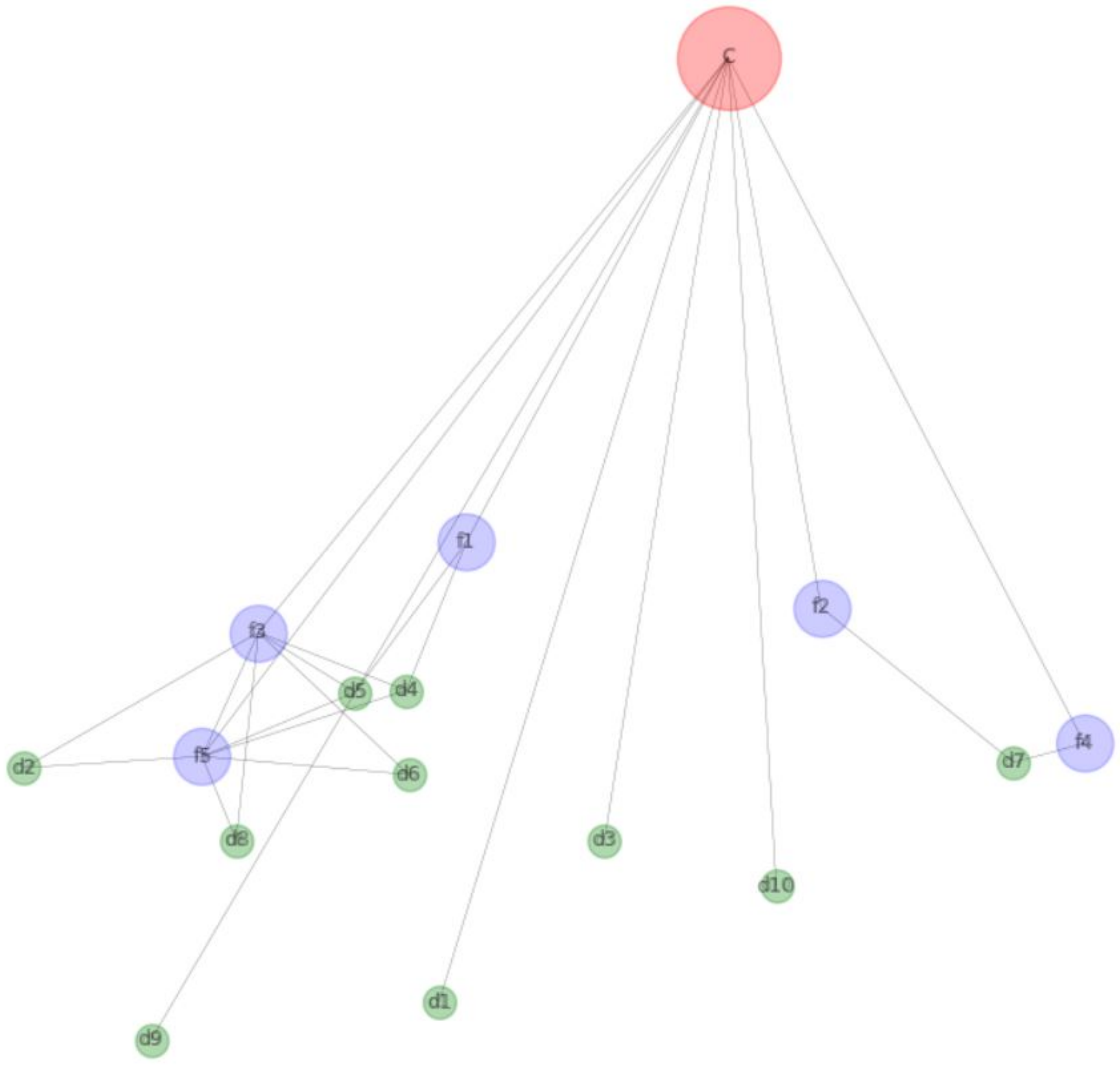


Figure 3.6: Simulation with 3 Fog nodes and 15 Edge devices

As mentioned before, every time we run the simulation, the Fog nodes and the edge devices are generated in different positions with random attributes. Figure 3.6 shows an instance of our simulation with 5 Fog nodes and 10 edge devices, where the nodes are represented using color coded circles. The Cloud node in red, the Fog nodes in blue and the edge devices in green. As we can see, all Fog nodes are connected to the Cloud node, while the lines between the edge devices and the Fog nodes represent if a device is in the range of a Fog node. The actual links are determined after the aforementioned calculations, to get the optimal connections. The connections between Fog nodes and other Fog nodes show that a neighbor cluster was formed. In this particular case, it happened between nodes **f1** and **f3**. As we can see edge devices **d1**, **d3**, **d9**, and **d10** did not fall in the range of any of the Fog nodes, therefore they connected directly to the Cloud.

In the next chapter we present how we intend to assess the proposed solution and discuss the experiments done on our system, and analyze the results to get insights about its strong points and weaknesses.

Chapter 4

Evaluation methodology, experiments and results

4.1 Outline

In this chapter, we state how we intend to evaluate the system, then we present the experiments done on the system and review the obtained results. Lastly, we highlight the strengths and shortcomings of our implementation.

4.2 Evaluation Methodology

To assess the performance of our implementation, we benchmark the overall process by running the simulation with different sets of numbers for the Fog nodes and the devices. Since the semantic reasoning and clustering directly affects the overall all performance, we can test for how the system scales by increasing the number of the Fog nodes and edge devices. Moreover, we compare between the two solvers we used by focusing on the differences in run-times, and the application response times obtained from the varied possible scenarios of how the application is being sent.

These scenarios are:

- The application is directly sent to the Cloud
- The application is sent to the Cloud through a Fog node
- The application is sent to a single Fog node
- The application is sent to an In-range cluster
- The application is sent to a neighbor cluster

We evaluate the utility of the two proposed clustering techniques, by comparing and the response times obtained from each method of sending applications.

4.3 Experiments and results

The machine specifications that the simulations are being executed on, will certainly have an effect on the performance of the system and the obtained results.

All tests where executed on a single machine with the following specifications:

- **Operating system**: Windows 10 Pro
- **CPU**: Intel Core i7-4710HQ @ 2.5 Ghz, base Clock Speed up to 3.5 Ghz Turbo Speed, 6MB L3 cache
- **Memory**: 16GB DDR3 @ 1600 MHz, Dual channel
- **Storage**: 500 GB NAND Sata SSD

Since the simulation generates random attributes for the Fog nodes, the edge devices and the connections, we have to run the simulation for number of times and calculate the means of the values to acquire decisive results. On one hand, for bench-marking and comparison between the run-times of the solvers, we ran the simulation using different sets of numbers for the Fog nodes and edge devices. For each set of numbers, we ran the simulation 50 times with the ILP solver, and 50 times with the FFHA. On the other hand, for the response time analysis, we run the simulation with the same number of Fog nodes and edge devices 50 times. In that manner, we focus on how the clustering techniques affect the response times. Figure 4.1 shows an example of the simulation with 5 Fog nodes and 250 edge devices.

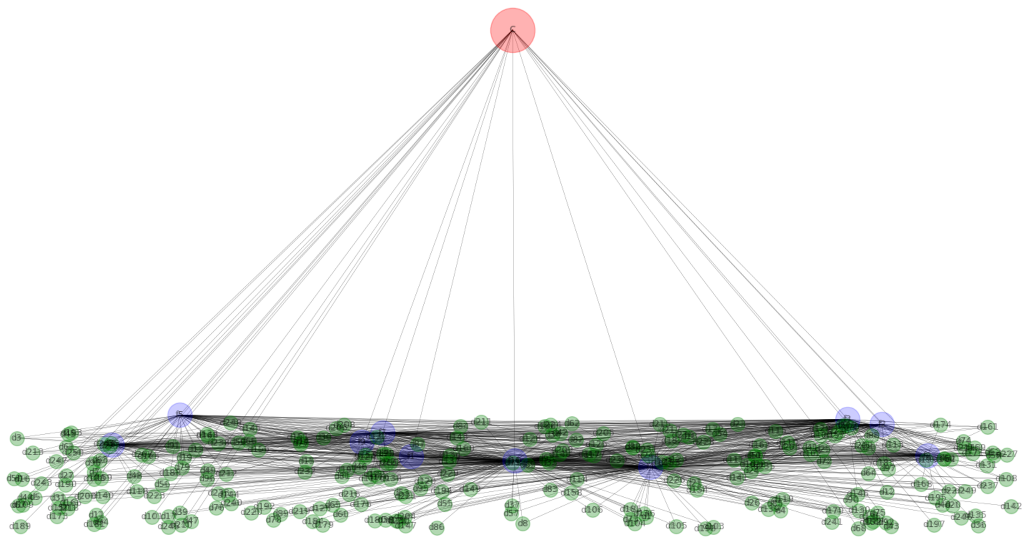
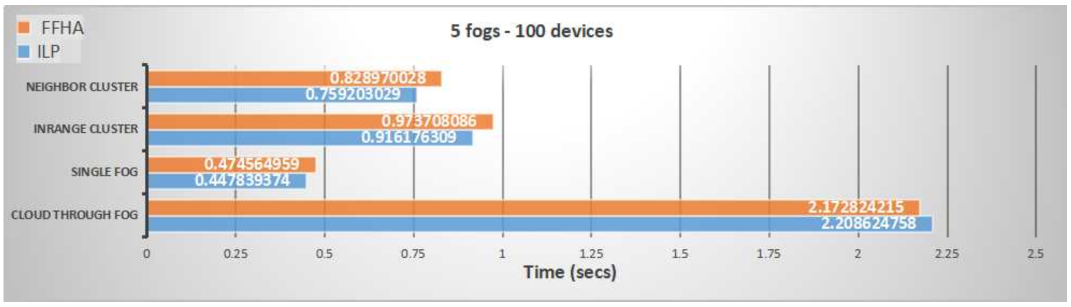


Figure 4.1: Simulation example of 10 Fog nodes and 250 devices

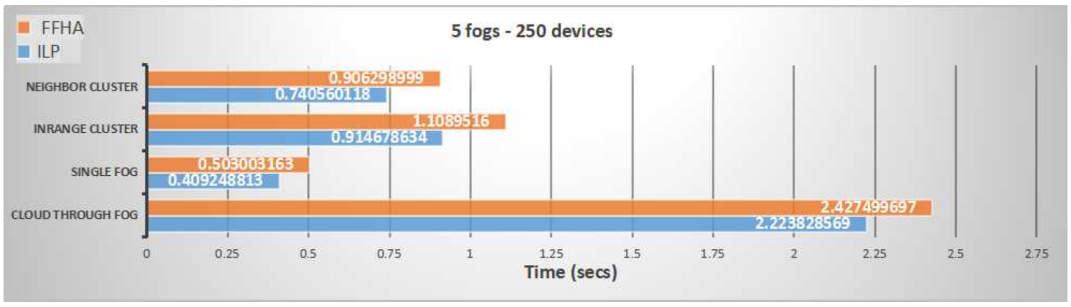
4.3.1 System performance

To evaluate the optimization techniques used to solve the task allocation problem, we focus on the response times of the applications in different scenarios, and we measure the average run-time for all the scenarios. We ran the simulation with the following sets of parameters 50 times with the ILP solver and 50 times with the FFHA.

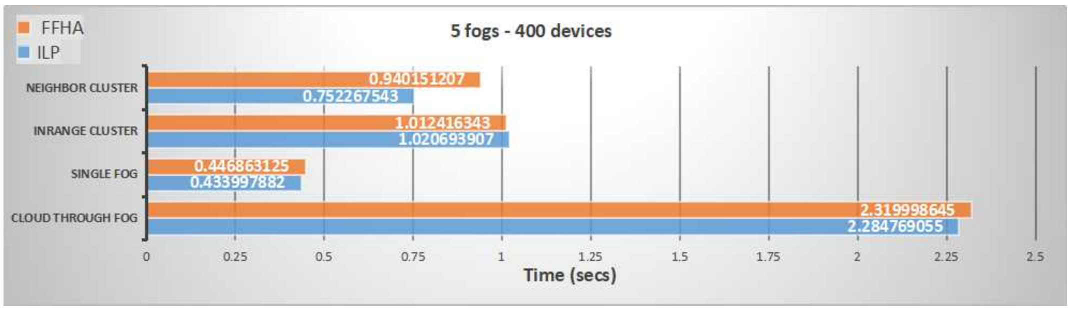
- **5** Fog nodes with 100, 250 and 400 edge devices
- **10** Fog nodes with 100, 250 and 400 edge devices
- **20** Fog nodes with 100, 250 and 400 edge devices



(a) 5 Fogs-100 devices



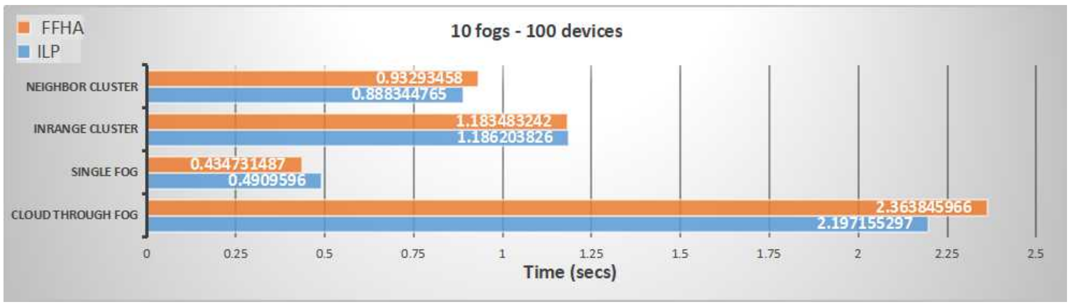
(b) 5 Fogs-250 devices



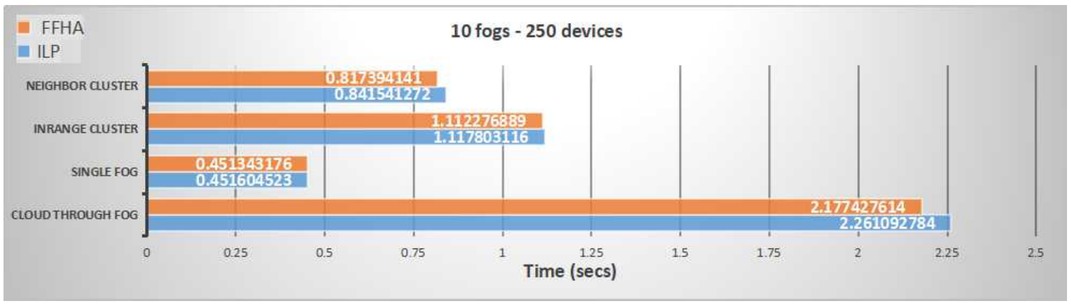
(c) 5 Fogs-400 devices

Figure 4.2: Response times for 5 Fogs and 100, 250, 400 devices (ILP - FFHA)

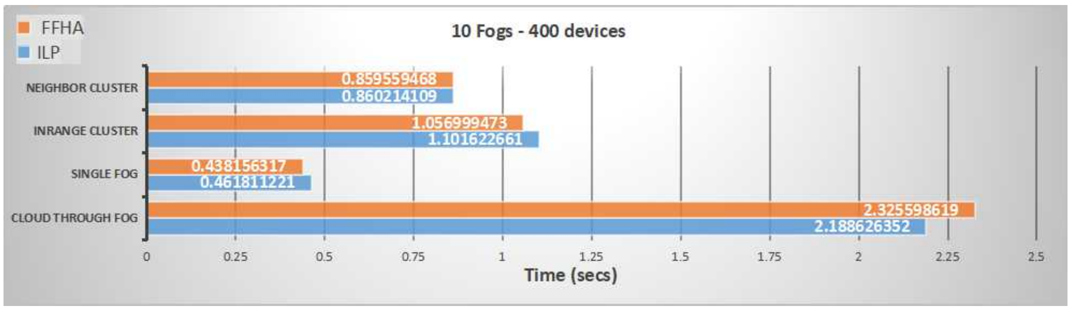
As expected, the results show that the ILP solver acquires slightly better response times than that of the FFHA, in the case of running the simulation with 5 Fog nodes and 100, 250, and 400 devices as shown in Figure 4.2.



(a) 10 Fogs-100 devices



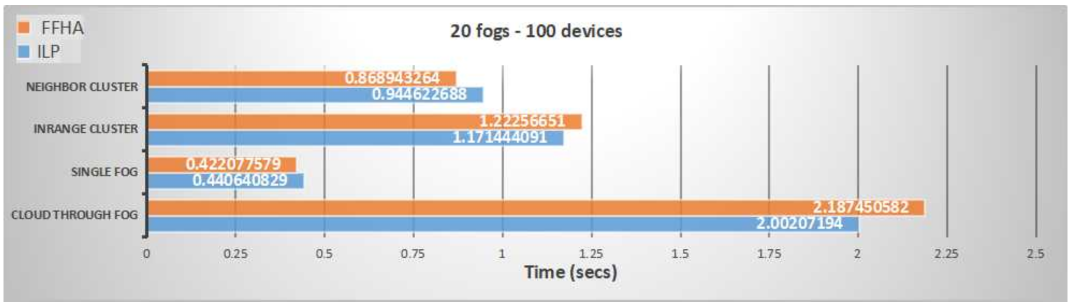
(b) 10 Fogs-250 devices



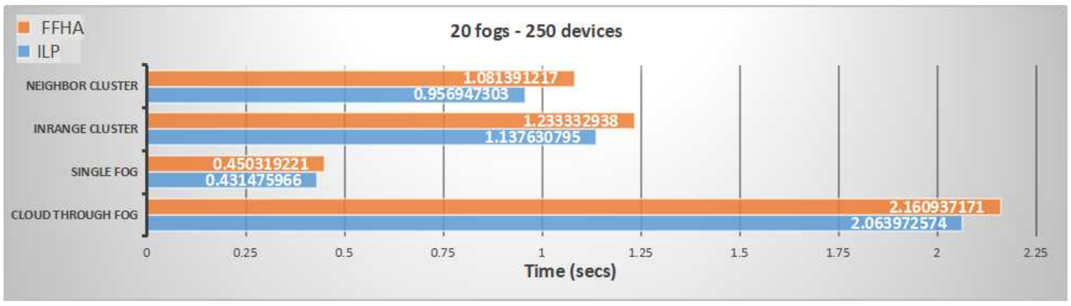
(c) 10 Fogs-400 devices

Figure 4.3: Response times for 10 Fogs and 100, 250, 400 devices (ILP - FFHA)

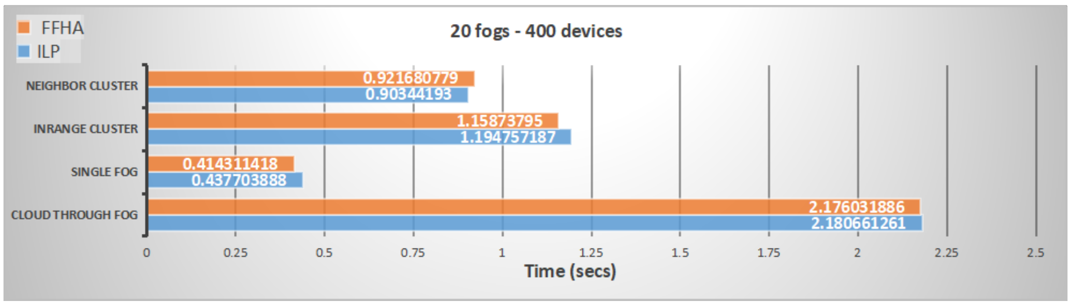
When running the simulation with 10 Fog nodes and the same sets of devices, the results show that in some cases, the FFHA gets better response times compared to the ILP solver as shown in Figure 4.3.



(a) 20 Fogs-100 devices



(b) 20 Fogs-250 devices



(c) 20 Fogs-400 devices

Figure 4.4: Response times for 20 Fogs and 100, 250, 400 devices (ILP - FFHA)

The Figure 4.4 shows the results when running the simulation with 20 Fog nodes. We can see that the results are similar to running the simulation with 10 Fog nodes. Where the ILP solver performs slightly better than the FFHA, yet the difference is not significant.

Overall, regarding the application response times, given the different methods of sending the application. The results gathered from running the simulation with the aforementioned sets of numbers for the Fog nodes and the devices, we can deduce that the ILP solver has a slight edge over the FFHA. However, the difference is rather minor. That is due to the fact that we perform the memory checks mentioned in Chapter 3, as an initial filter, then the solvers optimize where the tasks are processed to get the least possible response time.

In contrast, when reviewing the run-times of the simulations, there is a substantial difference between the performance of the ILP solver and the FFHA, where the FFHA performs much faster than the ILP solver as shown in Figure 4.5.

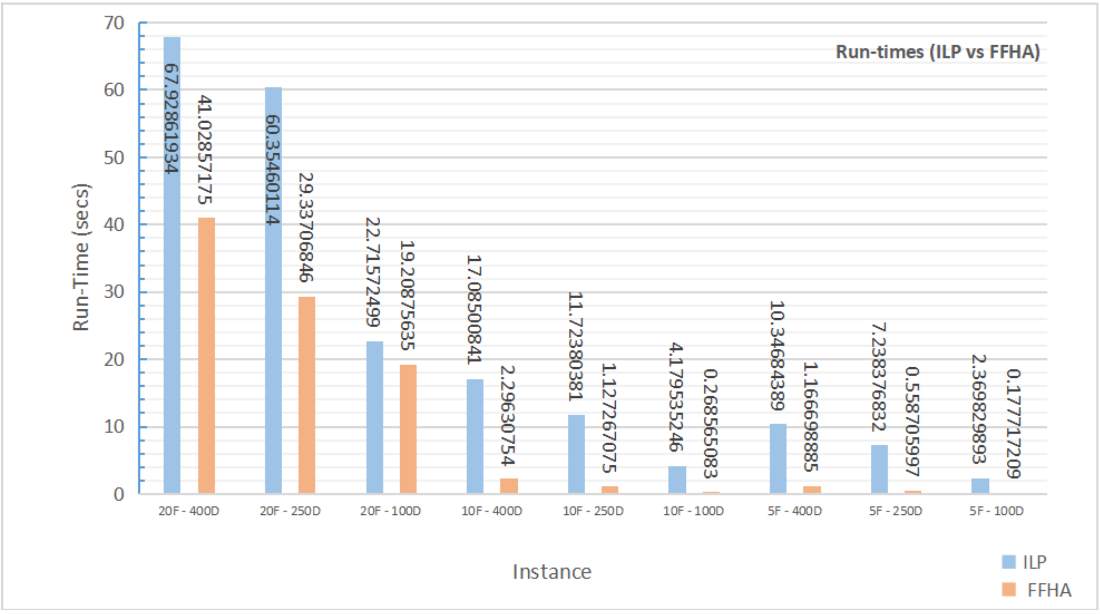


Figure 4.5: Run-times (ILP vs FFHA)

On one hand, increasing the number of edge devices linearly affects the run-time. On the other hand, we can see an exponential increase in run-time when increasing the number of Fog nodes, which is unsurprising, since this increases the number combinations when forming either an in-range or neighbor cluster.

To measure the consistency of the optimization techniques, we calculate the standard deviation, of both the response times and the run-times by conducting another experiment, using an instance size of 10 Fog nodes and 100 devices, and 100 iterations. Additionally, we calculated the number of Fog nodes in the clusters from the same instance, to give insight on the size of the clusters. Tables 4.1 and 4.2 show the average response times and run-times with their corresponding Standard Deviations (STD), when running the simulation using the ILP solver and the FFHA respectively.

ILP		
	Avg response times	**Response times STD**
Cloud direct	17.1991722	8.71893098
Cloud through Fog	2.18383078	0.23542268
Single Fog	0.43523073	0.10711002
In-range cluster	1.09988548	0.21959514
Neighbor cluster	0.88225381	0.21159424
	Avg run-time	**Run-time STD**
	3.82993240	0.81419498

Table 4.1: ILP - Averages and Standard deviations of response times & run-time

FFHA		
	Avg response times	**Response times STD**
Cloud direct	18.2210832	10.5720824
Cloud through Fog	2.19942492	0.26223306
Single Fog	0.45561651	0.11327142
In-range cluster	1.09464597	0.19415273
Neighbor cluster	0.93392979	0.22757457
	Avg run-time	**Run-time STD**
	0.22792500	0.09110830

Table 4.2: FFHA - Averages and Standard deviations of response times & run-time

When comparing the run-times STD, we can see that the FFHA behaves more consistently than the ILP solver, with less deviation.

Table 4.3 presents the average number of Fog nodes in both the in-range clusters and the neighbor clusters. We can see that the numbers are marginally higher when using the FFHA, which indicates that when using the ILP solver, we obtain clusters containing less Fog nodes. Conclusively, the results reflect our choice of Fog node and application properties.

Cluster size		
Type	**ILP**	**FFHA**
In-range cluster	2.713035699	2.912991401
Neighbor cluster	3.294254702	3.50243713

Table 4.3: Average cluster size

4.3.2 Utility of Fog clustering

We focus mainly on two aspects to evaluate the utility of our proposed clustering techniques. First, we analyze the overall response times gathered from the different methods of sending the application, regardless of the optimization technique. Next, we consider the number of devices being served by the Fog layer when no clustering technique is applied, using only in-range clustering, and using both in-range and neighbor clustering.

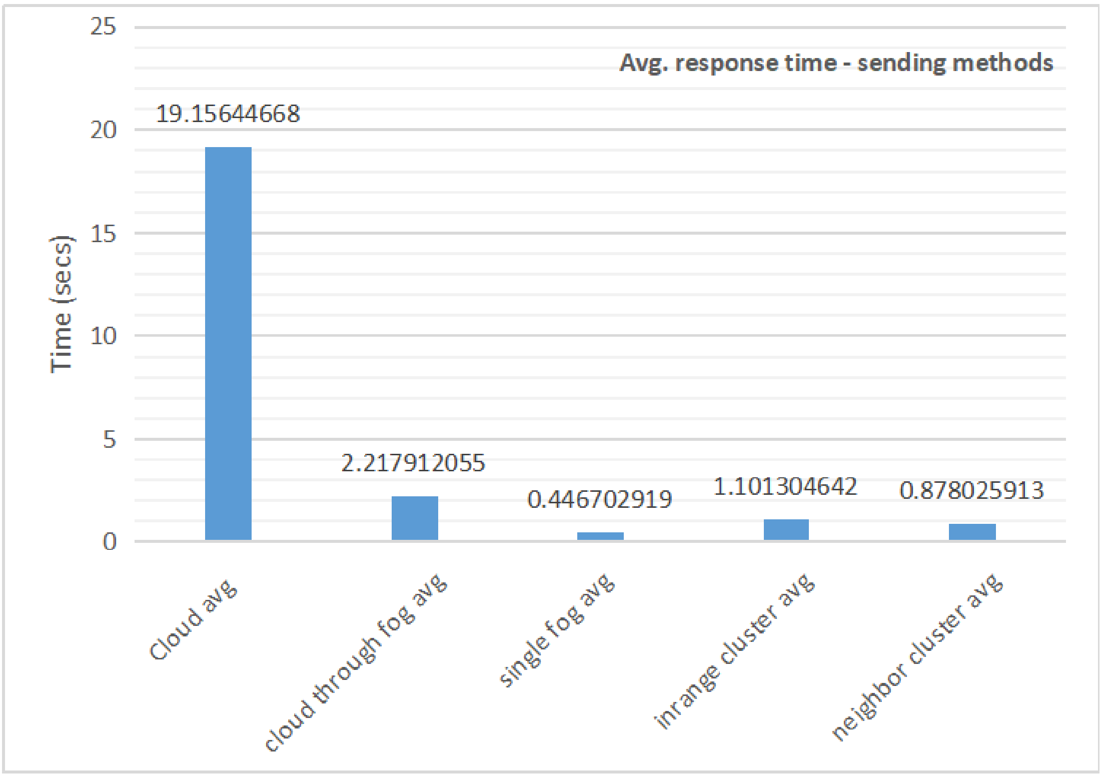


Figure 4.6: Average response times of different methods of sending applications

Figure 4.6 shows the difference in application response times when sending the application using different methods. As we can see, the difference between sending the application directly to the Cloud and utilizing the Fog paradigm is huge, which is expected due to the physical distance and number of hops between the devices and the Cloud. Moreover the data rate of the connection is much slower compared to the ones between the Fog nodes and the devices. Processing an application in a single Fog node provides the fastest response time, since no task allocation optimization takes place, and also because that application is just being sent over one link.

Our proposed clustering techniques deliver relatively fast response times, while enabling the service of a larger number of applications, with more demanding requirements. Figure 4.7, presents the percentages of applications being served regarding the different scenarios of how the applications are being sent and processed; directly to the Cloud, Cloud through a Fog node, a single Fog node, in-range clusters and neighbor clusters. We obtained the percentages shown in Figure 4.7, by running the simulation using 10 Fog nodes and 100 edge devices for 100 iterations, without any clustering techniques, with only the In-range clustering, and with both In-range and neighbor clustering. The results show a significant decrease in applications being sent to the Cloud when applying the clustering techniques, which in turn results in faster response times overall.

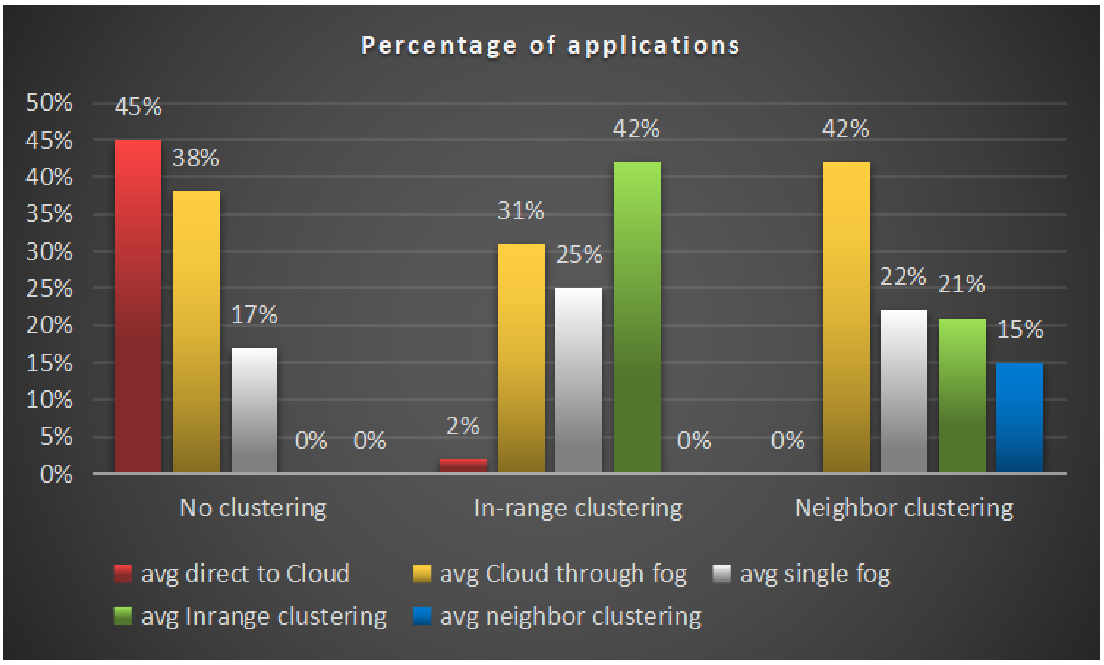


Figure 4.7: Percentage of applications sent using different methods

4.4 Discussion

Our implementation shows that there is a massive gain in response times when utilizing the Fog computing paradigm, compared to just relying on the Cloud.

After reviewing the results, it is apparent that the application response times are fundamentally faster when utilizing Fog nodes or clusters. While it is faster to process an application in a single Fog node, than to divide its tasks among several nodes within a cluster, some applications are more demanding and will require more resources than those of a single Fog node. Although the response times are slower when processing an application in a cluster, our implementation addresses the issue of processing more demanding applications in the Fog layer. In our simulation, we just considered one type of application profile, while in real-life scenarios, we have to consider a variety of application and task types, which in consequence will behave differently, and give varying results. Regrading the scalability of our implementation, one notable observation, is that the system starts experiencing crashes when simulating the system with more than 30 Fog nodes and 2000 edge devices. Furthermore, the values of the response times gathered from the results, where the application is sent to an in-range cluster or a neighbor cluster, do not support the real-time response requirement [42]. The reason being, that in our simulation we run all processes on one local machine. In a real-life test environment, the semantic reasoning and task allocation optimization should be processed in each of the Fog nodes, which will potentially reduce the response times.

Chapter 5

Conclusions and Future work

5.1 Conclusions

Fog computing is a very promising paradigm, and coupled with 5G broadband network they could prove essential in realizing the next stage of digital evolution. Its main role is to take some of the load off the Cloud, either by processing entire applications or by pre-processing the data, then sending it to the Cloud. However, there are several of its aspects need to be addressed for it to be adopted in real-life infrastructures. One of those aspects is related to the hardware limitations of the Fog devices. With edge devices progressively incorporating more powerful hardware, the requirements of their requests, grows correspondingly. Clustering of Fog devices could feasibly address this issue. In this thesis, we proposed two dynamic clustering techniques for Fog nodes, using a semantic description of their resources and the application's requirements to answer the research questions, presented in Chapter 1. The first technique involves clustering the Fog nodes that are in range of an edge device, while the other technique incorporates the clustering of Fog nodes in the vicinity of other Fog nodes. Additionally, we modeled and simulated the C2F environment using Python to test different scenarios. However, the clustering of Fog nodes leads to another concern, namely, the task allocation problem. In order to manage the issue that we modeled as an optimization problem, we used the PuLP python package to implement an ILP solver. This solver finds the optimal solution, but its run-time is greatly affected by the number of Fog nodes and edge devices. Consequently, we implemented a FFHA to minimize the effect of instance size on run-time. We then ran several tests against the system to evaluate its performance. The experiments showed that applying the FFHA achieved similar results to those of the ILP solver, but with a major decrease in run-time. Furthermore, we showed the difference in response times when sending applications directly to the Cloud, and to either a single Fog node, an in-range cluster, or a neighbor cluster. We also assessed the utility of Fog clustering, by comparing the number of applications being sent directly to the Cloud when no clustering method is applied, when only applying in-range clustering, and when applying both in-range and neighbor clustering. The results suggest that Fog node clustering enables more demanding applications to be processed in the Fog layer, which in turn reliefs the Cloud from some its workload.

To conclude, we acknowledge that this thesis work has accomplished the proposed objectives mentioned in Chapter 1, through the demonstrated research, implementation and the results obtained from analyzing the outcomes of the simulations.

5.2 Future work

While our proposed implementation presents and evaluates two clustering techniques for Fog nodes, as regard to the scope of the project, evidently, there is still more work to be done in the field. Our implementation is executed on one local machine, which leaves room for a more realistic testing by employing real machines for the Cloud, Fog nodes and edge devices. Additionally, job handling, scheduling and queuing are key factors in developing a fully functional system. By utilizing real-life data sets obtained from the increasing number of connected devices generating, processing, and sending heterogeneous data, could lead to more realistic results. Finally, security is undoubtedly a major concern that can not be overlooked. The increased amount of connected devices and Machine to Machine (M2M) communication, potentially raises more security vulnerabilities that can be exploited.

5.3 Abbreviations

- **IoT**: Internet of Things
- **IoE**: Internet of Everything
- **QoS**: Quality of Service
- **LoS**: Line of Sight
- **MDC**: Micro Data Center
- **C2F**: Cloud to Fog
- **STLS**: Smart Traffic Light System
- **ECG**: Electrocardiogram
- **WoT**: Web of Things
- **MILP**: Mixed Integer Linear Programming
- **ILP**: Integer Linear Programming
- **CBC**: Coin-or Branch and Cut
- **LP**: Linear Programming
- **WSGA**: Weighted Sum Genetic Algorithm
- **NSGA-II**: Non-dominated Sorting Genetic Algorithm II
- **MOEA/D**: Multi Objective Evolutionary Algorithm based on Decomposition
- **FFHA**: First Fit Heuristic Algorithm

- **MIPS**: Million Instructions per Second
- **RAM**: Random Access Memory
- **DR**: Data Rate
- **MPS**: Mathematical Programming System
- **M2M**: Machine to Machine
- **STD**: Standard Deviation

References

[1] IHS Markit Ltd. IoT platforms: Enabling the Internet of Things. *White Paper*, pages 4–5, 2016.

[2] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.*, 29(7):1645–1660, 2013.

[3] Harshit Gupta, Sandip Chakraborty, Soumya K. Ghosh, and Rajkumar Buyya. Fog computing in 5g networks: an application perspective. 2017.

[4] Mahbuba Afrin, Md Mahmud, and Md. Abdur Razzaque. Real time detection of speed breakers and warning system for on-road drivers. *Test*, pages 495–498, 12 2015.

[5] Noah Shachtman. Feds look to fight leaks with 'fog of disinformation', July 2012. Accessed 2020-02-12.

[6] Jonathan Bar-Magen. Fog computing: introduction to a new cloud evolution. 2013.

[7] Cisco Systems. Cisco delivers vision of fog computing to accelerate value from billions of connected devices, January 29 2014. Accessed 2020-02-14.

[8] Flavio Bonomi, Rodolfo A. Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. *Test*, pages 13–16, 2012.

[9] Raed Shubair and Fatima Al-Ogaili. Millimeter-wave mobile communications for 5g: Challenges and opportunities. 06 2016.

[10] Rainer Schmidt, Michael Möhring, Ralf-Christian Härting, Christopher Reichstein, Pascal Neumaier, and Philip Jozinović. Industry 4.0 - potentials for creating smart products: Empirical research results. pages 16–27, 2015.

[11] Fatos Xhafa, Burak Kilic, and Paul Krause. Evaluation of iot stream processing at edge computing layer for semantic data enrichment. *Future Generation Computer Systems*, 105, 12 2019.

[12] Stefan Lämmer and Dirk Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(04):P04019, apr 2008.

[13] Roxanne Hawi, George Okeyo, and Michael Kimwele. Techniques for smart traffic control: An in-depth review. *International Journal of Computer Applications Technology and Research*, 4:566–573, 07 2015.

[14] Tahere Royani, Javad Haddadnia, and Mohammad Alipoor. Control of traffic light in isolated intersections using fuzzy neural network and genetic algorithm. *International Journal of Computer and Electrical Engineering*, pages 142–146, 01 2013.

[15] Seiamak Vahid, Rahim Tafazolli, and Marcin Filo. Small cells for 5g mobile networks. pages 63–104, 05 2015.

[16] Rui Peng, Alexander Aved, and Kien Hua. Real-time query processing on live videos in networks of distributed cameras. *IJITN*, 2:27–48, 01 2010.

[17] Patrick Schneider and Fatos Xhafa. Data semantic enrichment for complex event processing over iot data streams. Master's thesis, UNIVERSITAT POLITECNICA DE CATALUNYA (UPC), 2019.

[18] Fano Ramparany, Fermín Galán, Javier Soriano, and Tarek Elsaleh. Handling smart environment devices, data and services at the semantic level with the fi-ware core platform. *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, 10 2014.

[19] Rustem Dautov, Iraklis Paraskakis, and Mike Stannett. Towards a framework for monitoring cloud application platforms as sensor networks. *Cluster Computing*, 17(4):1203–1213, 2014.

[20] Michael Compton, Payam M. Barnaghi, Luis Bermudez, Raul Garcia-Castro, Óscar Corcho, Simon J. D. Cox, John Graybeal, Manfred Hauswirth, Cory A. Henson, Arthur Herzog, Vincent A. Huang, Krzysztof Janowicz, W. David Kelsey, Danh Le Phuoc, Laurent Lefort, Myriam Leggieri, Holger Neuhaus, Andriy Nikolov, Kevin R. Page, Alexandre Passant, Amit P. Sheth, and Kerry Taylor. The SSN ontology of the W3C semantic sensor network incubator group. *J. Web Semant.*, 17:25–32, 2012.

[21] Dave Raggett. The web of things: Challenges and opportunities. *IEEE Computer*, 48(5):26–32, 2015.

[22] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Optimized iot service placement in the fog. *Service Oriented Computing and Applications*, 11(4):427–443, 2017.

[23] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource manage-

ment techniques in internet of things, edge and fog computing environments. *CoRR*, abs/1606.02007, 2016.

[24] Redowan Mahmud, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya. Profit-aware application placement for integrated fog-cloud computing environments. *J. Parallel Distributed Comput.*, 135:177–190, 2020.

[25] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Optimized iot service placement in the fog. *Service Oriented Computing and Applications*, 11(4):427–443, 2017.

[26] Isaac Lera, Carlos Guerrero, and Carlos Juiz. YAFS: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758, 2019.

[27] Vitor Barbosa C. Souza, Xavier Masip-Bruin, Eva Marín-Tordera, Sergio Sánchez-López, Jordi Garcia, Guang-Jie Ren, Admela Jukan, and Ana Juan Ferrer. Towards a proper service placement in combined fog-to-cloud (F2C) architectures. *Future Gener. Comput. Syst.*, 87:1–15, 2018.

[28] Brij B. Gupta, Yogachandran Rahulamathavan, Shingo Yamaguchi, Tyson Brooks, and Zheng Yan. IEEE access special section editorial: Recent advances in computational intelligence paradigms for security and privacy for fog and mobile edge computing. *IEEE Access*, 7:134063–134070, 2019.

[29] Eva Marín-Tordera, Xavier Masip-Bruin, Jordi Garcia Almiñana, Admela Jukan, Guang-Jie Ren, and Jiafeng Zhu. Do we all really know what a fog node is? current trends towards an open definition. *Comput. Commun.*, 109:117–130, 2017.

[30] Rustem Dautov and Salvatore Distefano. Automating iot data-intensive

application allocation in clustered edge computing. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, 06 2019.

[31] Oracle Linux. Linux containers, 2019. Accessed 2020-03-08.

[32] Lucy Hattersley. Raspberry pi 4, 3a+, zero w - specs, benchmarks '—&' thermal tests, 2019. Accessed 2020-03-15.

[33] NGMN Alliance. 5g white paper-executive version. *White Paper, December*, 2014.

[34] Paul Farrell and Hong Ong. Communication performance over a gigabit ethernet network. pages 181 – 189, 03 2000.

[35] Afaq Khan, Mohammed Qadeer, Juned Ansari, and Sariya Waheed. 4g as a next generation wireless network. *Future Computer and Communication, International Conference on*, 0:334–338, 04 2009.

[36] Edward G. Coffman, J. Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: Survey and classification. 1-5:455–531, January 2013.

[37] R. Yesodha and T Amudha. A comparative study on heuristic procedures to solve bin packing problems. *International Journal in Foundation of Computer Science '—&' Technology (IJFCST)*, 2:37–49, 11 2012.

[38] A. Makhorin. Glpk (gnu linear programming kit). Available at `http://www.gnu.org/software/glpk/glpk.html`.

[39] Coin-or (common infrastructure for operations research). Available at `http://www.coin-or.org`.

[40] Alberto Ceselli, Marco Fiore, Marco Premoli, and Stefano Secci. Optimized assignment patterns in mobile edge cloud networks. *Comput. Oper. Res.*, 106:246–259, 2019.

[41] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[42] J. Huang, Y. Wang, and F. Cao. On developing distributed middleware services for qos- and criticality-based resource negotiation and adaptation. *Real-Time Systems*, 16:187–221, 05 1999.