Scalable and Robust Fog-Computing Design & Dimensioning in Dynamic, Trustless Smart Cities

by

Ismael Sanchez-Martinez

A thesis submitted for the degree of Doctor of Philosophy

April 2024

Université de Montréal

Department of Computer Science and Operations Research

Résumé

Le concept de Ville Intelligent concerne l'interconnectivité totale de plusieurs industries vers l'amélioration des modes de vie des résidents. Ceci est rendu possible par la croissance et l'utilisation généralisée de l'Internet des objets (IoT), un vaste réseau de dispositifs de collecte de données répartis dans de multiples applications. Cependant, la plupart des appareils IoT disposent de peu de ressources et s'appuient sur des serveurs externes pour traiter et stocker les données collectées. En raison de la congestion et de la distance élevées, les centres de données Nuage (Cloud) peuvent entraîner une latence élevée dans leur réponse IoT, ce qui peut être inacceptable dans certaines applications IoT. Au lieu de cela, l'informatique Brouillard (fog-computing) a été proposé comme une couche hétérogène hautement virtualisée de serveurs à la périphérie du réseau, ce qui permet un traitement des données IoT à faible latence.

Les contributions actuelles au brouillard informatique supposent qu'une infrastructure de brouillard est déjà en place. De plus, chaque contribution nécessite des caractéristiques différentes sur l'infrastructure du brouillard. Cette thèse formule un schéma de conception et de dimensionnement évolutif et modifiable pour une infrastructure de brouillard généralisée. Ceci est modélisé et résolu sous la forme d'un programme linéaire à nombres entiers mixtes (MILP), et détendu à l'aide de plusieurs techniques telles que la génération de colonnes et la décomposition de Benders. De nombreuses préoccupations concernant les performances du réseau brouillard sont prises en compte et résolues, telles que le trafic IoT élevé, la congestion du réseau et les dysfonctionnements des nœuds brouillard. Les nœuds de brouillard dynamiques, tels que les nœuds de brouillard à la demande et les véhicules aériens sans pilote mobiles (UAV-brouillard) sont intégrés dans les modèles de conception et de dimensionnement actuels pour ajouter de la flexibilité et de la robustesse au réseau. Un système basé sur la blockchain et des preuves de connaissance nulle est introduit pour renforcer l'intégrité des nœuds de brouillard. Le résultat est un schéma de conception et de dimensionnement évolutif pour une infrastructure de brouillard robuste, flexible et fiable dans un environnement de brouillard-IoT dynamique et malveillant.

Mots Clés — Fog-computing, Internet des objets, Conception et Dimensionnement, Décomposition de Benders, Génération de Colonnes, Véhicules Aériens sans Pilote, Blockchain

Summary

The concept of a Smart City relies on the full interconnectivity of several industries towards the amelioration of resident lifestyles. This is made possible by the growth and wide-spread use of the Internet of Things (IoT) – a large network of data collection devices throughout multiple applications. However, most IoT devices have few resources, and rely on external servers to process and store the collected data. Due to high congestion and distance, Cloud data centres may cause high latency in their IoT response, which may be unacceptable in certain IoT applications. Instead, fog-computing has been proposed as a highly-virtualized heterogeneous layer of servers on the network edge, resulting in low-latency IoT data processing.

Current contributions in fog-computing assume a fog infrastructure is already in-place. Furthermore, each contribution requires different characteristics on the fog infrastructure. This thesis formulates a scalable and modifiable design & dimensioning scheme for a generalized fog infrastructure. This is modeled and solved as a mixed-integer linear program (MILP), and relaxed using several techniques such as Column Generation and Benders Decomposition. Many concerns on the fog network performance are considered and addressed, such as high IoT traffic, network congestion, and fog node malfunctions. Dynamic fog nodes, such as on-demand fog nodes and mobile fog-enabled unmanned aerial vehicles (fog-UAVs) are integrated into current design & dimensioning models to add flexibility and robustness to the network. A system based on blockchain and zero-knowledge proofs is introduced to enforce integrity on the fog nodes. The result is a scalable design & dimensioning scheme for a robust, flexible, and reliable fog infrastructure in a dynamic and malicious IoT-fog environment.

Keywords— Fog-computing, Internet of Things, Design & Dimensioning, Benders Decomposition, Column Generation, Unmanned Aerial Vehicles, Blockchain

Contents

Acknowledgement

This doctoral thesis is the culmination of years of work, and would not have been possible without the support and guidance from several key individuals.

Firstly, I am very fortunate to have worked with my two brilliant supervisors: Prof. Abdelhakim Senhaji Hafid, and Prof. Michel Gendreau. They have consistently challenged my knowledge, and pushed my limits to improve as a researcher. Most importantly, they believed in me, and had faith in my ability to complete this PhD and continue with future endeavors. I am grateful for their patience, insights, and support.

I would like to acknowledge the Natural Sciences and Engineering Council of Canada (NSERC) for the financial support of my PhD. The NSERC scholarship allowed me to have academic freedom to take my time with my work. Furthermore, I would like to thank Prof. Hafid for coaching and preparing me to qualify for this key scholarship, and for helping me secure the next step of my academic career post-PhD.

I would like to thank my colleagues and students at UdeM who have created a welcoming environment for me at the university. It was heartwarming to be part of a community of people with a common goal, and they motivated me to never give up.

Finally, I would like to extend my sincere appreciation to my closest friends and family who have stayed by my side throughout my studies. You have been there to lift me up when I'm down, to brighten my days, and to celebrate every milestone of my studies. This accomplishment would not have been possible without you.

Chapter 1

Introduction

The term *smart city* is used to describe urban environments with a high degree of interconnectivity between wide-spread collected sensor data and urban industries [1]. Through the collaborative integration of information and communication technologies, smart cities aim to streamline city services and hence to ameliorate the lives of residents. Examples of participating industries in smart cities are smart agriculture, smart transportation, smart health and well-being, smart waste management, smart water management, and smart power grids. An efficient smart city application can choose the scale and portion of affected residents and structures through smart monitoring and system modularization. This monitoring can cover structures such as homes, schools, offices, factories, and vehicles [1].

1.1 Internet of Things

Smart cities are enabled by data collectors and processors linked by network communication technology. The data collection is carried out by *things* and *smart objects* – singular and compound devices with intelligent interfaces that interact with the physical world [2]. We refer to the collaborative network of things and smart objects as the *Internet of Things* (IoT). We henceforth refer to the collection of things and smart objects as IoT devices.

Key performance features of IoT devices are:

- **Awareness** – to sense, interpret or react to events in the physical world;
- **Representation** – the use of programming models to process environmental data;
- **Interaction** – the ability to process control (inputs) or feedback (outputs) from users.

Physically, IoT devices are resource constrained, having only a limited amount of computer processing capabilities and internal storage. As a result, they rely on communication technology to quickly forward collected data for processing and long-term storage by external servers [2, 3].

Up until recently, IoT devices have relied on centralized Cloud mega data centres for processing and storage support. As IoT applications become more commonplace, the number of IoT requests being sent out to the Cloud is increasing year-over-year [4]. The rate of growth of IoT traffic towards Cloud causes internet congestion near the Cloud, and processing delays from the Cloud.

Furthermore, Cloud data centres are few in number, located physically worldwide. Hence, the distance alone may cause significant delays in IoT request transmission.

Certain IoT applications are latency-sensitive, i.e., require near-immediate data transmission, processing, and response. A small delay in milliseconds from external processors can result in noticeable decrease in application efficiency. Examples of latency-sensitive IoT applications are heart monitors and vehicular traffic status sharing. In such cases, we look towards processing IoT requests much closer to IoT devices, as to minimize computational latency.

1.2 Fog-computing

In response to high latency, the fog-computing paradigm has been proposed as highly virtualized "micro data centres" or "mini-Clouds" on the network edge [5, 6], i.e., in proximity to IoT devices for quick responses. Fog nodes are geographically distributed computing devices that may act as an intermediary between IoT and Cloud. That is, a fog node may process an IoT request in real-time, and forward the IoT data to the Cloud for long-term storage. Key characteristics of fog are as follows:

- **Real-time** – being geographically distributed on the network edge ensures real-time or low-latency communication with IoT.
- **Micro data centres** – typically have more processing power and storage than IoT devices, and less than Cloud.
- **Heterogeneous and numerous** – the fog layer is a large network of fog nodes with various computing, storage, access, and mobility capabilities.
- **Predominantly wireless accessibility** – enables wide-spread communication of IoT-fog, fog-Cloud or fog-fog communication.
- **High virtualization** – allows for constant and seamless service to mobile applications moving between fog node service regions.
- **Decentralized ownership** – fog nodes need not be owned by any one authority.
- **Interoperable** – heterogeneous fog nodes can collaborate to provide services that require information from different domains and providers.

In short, any device with communication, computing and storage capabilities can be a fog node [7].

Unlike Cloud data centres, fog devices are decentralized and geographically distributed, allowing for IoT connectivity with minimal processing latency. Fog nodes are far more numerous and predominantly wireless. However, they each have much fewer available computing and storage resources than Cloud data centres. A comparison of fog and Cloud layers is shown in Table 1.1.

It is important to note that fog nodes are not a replacement for Cloud data centres, but rather are meant to complement them. Indeed, IoT-fog-Cloud cooperation is essential in the realization of smart city technologies.

Feature	**Fog**	**Cloud**
Latency	Low	High
Distribution	Geographically distributed	Centralized
Distance from network edge	Close	Far
Number of nodes	Millions	Thousands
Resource quantity	Small to medium	Large
Access	Predominantly wireless	Wired or wireless
Heterogeneity	High	Low
Interaction with IoT	Real-time	Batch processing
Owned and managed	Various service providers	Few large organizations

Table 1.1: Comparison between the fog-computing and Cloud-computing layers

1.3 State of the Art - Fog-Computing

The development of our research is inspired by the strengths of previous contributions, and motivated by their limitations [8]. In our literature review, we focus on contributions related to fog architecture, resource management, security and integrity.

1.3.1 Role of Fog in Smart Cities

The prevalence of IoT aims to connect multiple industries towards the actualization and future of smart cities. The typical data collection strategy for resource-constrained IoT devices is to immediately forward any collected data to external sources. Sending this data directly to the Cloud may result in unnecessary network congestion and Cloud storage wastage from unfiltered IoT data. Instead, intermediary fog nodes can pre-process IoT data, forward only the relevant pieces to the Cloud, and return an associated response to IoT. Overall, this is a more efficient data sharing and data processing system [1].

IoT devices are often depicted as having minimal resources, whereas Cloud is often assumed to be able to handle any processing and storage task without restriction. That is, IoT devices and Cloud mega data centres represent the two extremes of the resource quantity spectrum. Fog resource capabilities can be ‘anything in-between’. Fog nodes could be small user-provided devices such as smart phones, internet routers and home networks [9], or larger servers capable of supporting neighourhood-, community-, or city-wide IoT application [10].

In large smart city applications, the fog layer can be interpreted as multi-layered, separating fog virtualization into layers based on fog resource quantity [10] or by proximity to IoT [11, 12]. Similarly, the fog layer can also be represented as a group of fog clusters based on proximity to each other [13]. The representation of fog is highly dependent on the requirements of the smart city application.

The general fog architecture uses a single fog layer of heterogeneous nodes that stretches from IoT to Cloud [7, 14, 15]. This is the most flexible interpretation of the fog layer, as it allows for fog nodes of various resource capacities and distances from IoT to collaborate to service IoT. This flexible architecture can be partitioned into any multi-tier or clustered architecture as needed. The focus of our research on the fog-computing layer is meant to be as flexible and general as possible. Indeed, a flexible geographically distributed single layer architecture makes full use of the strengths of fog-computing.

1.3.2 Resource management

Resource management in fog-computing refers monitoring available fog resources and assigning IoT requests to fog nodes accordingly. Fog resource management can be divided into three categories: 1) resource provisioning and allocation, 2) fog frameworks, and 3) data migration.

Resource provisioning and allocation

Resource *provisioning* refers to reserving resources within a fog node for future IoT processing, while resource *allocation* refers to the assignment of IoT requests to fog nodes. Clearly, these two processes work hand-in-hand. Resource provisioning schemes either exhibit prior provisioning – whereby resources are reserved based on projected IoT traffic [5], or on-demand provisioning – whereby resources are reserved as the IoT requests come in [16, 17]. Resource allocation schemes either exhibit prompt allocation – whereby IoT requests are allocated immediately to reserved fog resources [5, 17], or small batch allocation – whereby IoT requests are collected into small batches and processed together [16].

Fog orchestration

Fog orchestration refers to the automated configuration, management and coordination of fog resources. A *fog framework* facilitates fog resource orchestration policy. Clearly, the particular resource provisioning and allocation scheme of a fog application may be implemented by a fog framework. Certain IoT applications subdivide each request into multiple tasks that require a different service [18]. In this case, a fog framework would manage the appropriate distribution or IoT tasks among fog nodes. Fog frameworks may use a separate controller to monitor and control resource management [19, 20], or allow fog nodes to communication with each other via APIs [21]. Using a separate controller requires additional hardware in the fog layer, and inherently forms a clustered fog architecture, while an API-based framework requires additional software within fog nodes and communication latency between fog nodes.

Data migration

Data migration in fog refers to the transfer of IoT data between fog nodes. If an IoT device is connected to a specific fog node, then a migration may take place in either of two cases:

(a) Mobile IoT devices or mobile fog nodes move out of range of each other [22];

(b) A fog node malfunctions, becomes overloaded, or is corrupted, rendering it unusable [23].

There are numerous options for the implementation of resource management within the IoT-fog environment, each with its own additional hardware and software requirements. It is important to note that no particular resource management scheme is superior in every way. That is, every implementation has its strengths and weaknesses. Hence, the resource management of fog nodes should depend on the needs of fog nodes and IoT devices for a particular IoT-fog application.

1.3.3 Security and integrity

Security is of utmost importance in any communication system. The most important elements in IoT-fog security are *authorization* and *authentication* [24].

Authorization refers to the control and verification of data access, either others accessing your data or you accessing others' data. Authorization often takes the form of an *access control* policy – a separate table that defines everyone's access permissions. Authentication refers to verifying the identity of an individual. Authorization and authentication often work in tandem – we simultaneously authenticate someone's identity, and authorize the user for data access [25]. In a secure IoT-fog environment, authentication should be verified in both directions. That is, IoT devices and fog nodes should mutually authenticate each other prior to data sharing [26].

Once authorization and authentication have been established, IoT devices and fog nodes should share data via encrypted communication. In most contributions, standard encryption schemes such as RSA or elliptic curve cryptography (ECC) are used [26]. ECC is known to be more secure than RSA for equivalent key sizes [27]. However, the associated parameters of ECC need to be established by a trusted third-party.

Blockchain technology has been a focus of research in recent years in providing security in computer systems, include IoT [28]. A blockchain is a decentralized ledgers that exhibits block consensus, immutability, auditability and pseudo-anonimity across all blockchain nodes. Hence, they are viewed as a strong candidate for providing security in trustless environments [29]. Several contributions have proposed to use blockchain as the trusted third-party to facilitate and streamline mutual authentication and access control [30, 31]. Since fog and blockchain nodes are both distributed and decentralized, it is reasonable to combine them as *blockchain-enabled* fog nodes to further reduce communication latency [32].

Authentication and authorization work on the premise that a verified user will act fairly in all IoT-fog communications, data processing and data sharing. In a trustless IoT-fog environment, this cannot be guaranteed without additional integrity verification mechanisms. Data auditing techniques have been proposed to verify that fog nodes store correct, uncorrupted data from IoT [33, 34]. However, these techniques rely on fully cooperative fog nodes to adjust faulty data [34]. If a fog node is acting maliciously, then there is no guarantee of honest cooperation.

Up until now, no contributions have been made to verify the service integrity of fog nodes. That is, to ensure that fog nodes are processing IoT requests correctly, and responding to IoT with the correct data.

1.3.4 Fog design & dimensioning

Network design & dimensioning refers to the location (design) and resource quantities (dimensioning) of network nodes. Prior to our work, there existed only one contribution in the domain of fog design & dimensioning [35]. This contribution focused on the scalable design & dimensioning of fog nodes, road-side units (RSUs), and gateways to support autonomous vehicles. The optimal design & dimensioning configuration was based on a discrete set of vehicular traffic, where resource dimensioning was selected from a discrete pre-defined set of configurations. In reality, these sets are not likely known. Furthermore, the application is specific to Internet of Vehicles (IoV), and requires more overhead than a general fog infrastructure.

1.4 Motivation

Overall, current contributions in fog-computing study the interaction of IoT, Cloud and fog assuming the existence of a set fog infrastructure. There previously exists only one study on design &

dimensioning of a fog infrastructure, that is specific to IoV applications. There exists a need for a fog design & dimensioning scheme for a generalized fog infrastructure.

We observed the following limitations of the reviewed literature:

- Different contributions use a different IoT-fog-Cloud architecture. Specifying a multi-tier or clustered architecture neglects the inherent potential of using the entire distributed fog hierarchy (IoT-fog-Cloud continuum).
- Different contributions have different requirements for the additional hardware and software installed on fog nodes for optimal operations. That is, there is a lack of generality and heterogeneity in proposed fog systems.
- Current contributions do not consider the effects of overloaded or malfunctioning fog nodes. That is, resource management schemes are proposed under the assumption that fog nodes will always operate efficiently under a consistent and manageable IoT traffic load.
- Most resource management schemes operate on a 'one IoT request per fog node' system. That is, they do not consider that an IoT request may be partitioned into multiple tasks that require service from multiple fog nodes, which is the case for a number of IoT services.
- Current contributions assume all fog nodes are trustworthy and failure-resistant, and do not consider the possibility of malicious or unreliable fog nodes.

Every current application of fog requires a different type of fog infrastructure. We recognize the need for a design & dimensioning scheme that gives a generalized, and heterogeneous fog infrastructure. The largest strength and use case of fog-computing comes from its flexibility to fit any required IoT use case. Based on the summarized limitations, we consider a design & dimensioning scheme that constructs a heterogeneous single-layer fog infrastructure that considers traffic changes and node failures in a dynamic fog environment over multi-task IoT requests. The design & dimensioning scheme should be flexible and modifiable based on the particular requirements of the smart city application and resource management policy. It should also be *scalable* – to be tractable for very large networks of heterogeneous fog nodes and high IoT traffic.

Regarding fog integrity, current contributions in fog-computing rely on the assumption that fog nodes store and process IoT data correctly. Even if data is found to be corrupted, it is assumed to be either accidental or due to outside tampering. However, it is reasonable to consider that some fog nodes may act intentionally dishonestly [33]. There exists a need to enforce fog *integrity* – the honest and correct processing of fog nodes. Since IoT devices rely on fog nodes for service processing, it is imperative that fog nodes are kept accountable in providing fair and trusted service.

1.5 Research Objectives

The core research objective revolves around the scalable design & dimensioning of fog infrastructures where none exists, or the extensibility of current infrastructures in light of increasing IoT traffic. In addition, we develop a general blockchain-based system for enforcing security and integrity among fog nodes.

The proposed fog infrastructure will therefore provide low-latency responses, data security, and service integrity of IoT requests for a reliable, desirable, and widely available computing infrastructure.

1.5.1 Scalable Fog Design & Dimensioning for Dynamic Fog Infrastructure

In real-life IoT-fog applications, the number of incoming IoT requests is variable and sometimes volatile due to IoT mobility and infrequent periods of peak IoT traffic [36]. Potential failures in fog node hardware or IoT transmission may further cause stress on the fog infrastructure, and affect its reliability. Hence, we account for fluctuations in IoT traffic and and fog node failures in our proposed scheme.

Our fog design & dimensioning scheme accounts for fluctuations in a dynamic IoT-fog environment in three phases. (1) We define a fog design & dimensioning scheme for a general static fog infrastructure in a 'stable' IoT-fog environment. (2) We extend the static fog infrastructure to include on-demand fog nodes that activate in light of high IoT traffic and/or fog node failures. This design & dimensioning scheme results in a robust fog infrastructure. (3) We further extend the robust infrastucture to include mobile fog-enable unmanned aerial vehicles (UAVs) for flexible IoT service.

An effective fog infrastructure is able to support all incoming IoT requests with low-latency, regardless of the IoT traffic conditions. In certain situations, it is possible for IoT traffic to be 'bursty' ,i.e., to display infrequent periods of abnormally large number of IoT requests. Furthermore, it is possible for an IoT request to be composed of multiple tasks that can be serviced by multiple fog nodes [18]. Our research considers the worst-case scenario of IoT traffic. Therefore, we assume bursty multi-task IoT traffic.

1.5.2 Blockchain-based Security & Integrity Enforcement

In order to ensure fair service in the IoT-fog environment, it is crucial to hold fog nodes accountable if they do not process IoT requests as intended. That is, we enforce fog integrity through a blockchain-based service auditing system. Our system provides penalties for malicious fog nodes, and incentives for honest fog nodes. Furthermore, we streamline the mutual authentication and service payment processes within the blockchain-based system for a secure and fair IoT-fog environment.

1.6 Contributions

We seek to formulate a scalable design & dimensioning scheme for a general fog infrastructure in a volatile and trustless IoT-fog environment. Our first contribution focuses on the construction of a scalable and static fog infrastructure. That is, fog nodes are set in a fixed location, and cannot fail. The IoT environment is modeled as a set of IoT devices with requests of variable size, task partitioning, and arrival rate. Furthermore, we model the network congestion latency caused by high IoT traffic around each fog node, encouraging assignment of IoT requests to multiple fog nodes as opposed to just one. These considerations ensure the resulting fog infrastructure responds well to the worst-case IoT traffic and congestion patterns.

The static fog design & dimensioning scheme is developed based on a large percentile of probabilistic IoT data, and assumes fog nodes cannot fail. In the case the IoT data do surpass the given IoT threshold, or if any fog node fails while the network is near capacity, then the fog network will overload and cause large delays in IoT service. We propos to use on-demand fog nodes that will activate when the system is overloading. This robust design & dimensioning scheme focuses on the placement and resource dimensioning of the on-demand fog nodes, and defines a re-routing configuration for each static fog nodes towards an on-demand fog node. Both design & dimensioning models for static and robust fog infrastructure are solved with a mixed-integer linear program (MILP). They are further decomposed for scalability and solved with a column generation technique.

Assuming a large percentile of possible IoT traffic can result in a large fog infrastructure of static and dynamic fog nodes, many of which are unused a majority of the time. Indeed, depending on the distribution of IoT traffic, the upper end of the traffic distribution may be unlikely, but rather a worst-case scenario that should still be considered. Our third contribution considers the use of fog-enabled UAVs (fog-UAVs) to service high traffic or remote areas. An infrastructure of fog-UAVs would allow the service of the abnormally high IoT traffic that would normally overload the static and dynamic fog infrastructure. The design & dimensioning of fog-UAVs is formulated as a probabilistic location set-covering problem [37] and solved with anMILP under a relaxation approach and a Benders decomposition approach [38].

Once a fog infrastructure has been established, we look towards fortifying the security and integrity of all IoT-fog communication. We propose a blockchain-based system that enforces integrity of fog nodes through service auditing. The service audits are anonymously carried out by oracles, who use a zero-knowledge ring signature to prove their identity to the blockchain without revealing it to neither blockchain nor fog nodes. The system also enables secure mutual authentication and streamlined service payment from IoT to fog.

1.7 Key Assumptions

Several assumptions are made to simplify the formulated models towards the objectives of this thesis. The following key assumptions remain consistent throughout each chapter.

- Though IoT traffic can be volatile in the short-term, long-term traffic patterns are assumed to follow a Poisson distribution when influenced by Human Dynamics [36]. That is, a right-skewed distribution where IoT traffic may exhibit infrequent 'bursty' traffic spikes.
- Communication between IoT and fog does not consider latency produced by distance or the environment. When fog nodes are within a few hops from IoT, latency from distance is negligible. Hence, latency estimates focus on network congestion. However, in the case of direct communication, factors of environmental interference are not considered.

We recognize these assumptions must be validated in future work to verify the viability of the proposed fog implementation. Indeed, a future small-scale Proof of Concept (PoC) of a fog infrastructure will enable the study of IoT traffic and communication reliability in different environments. However, the current fog design & dimensioning scheme proposed in this thesis is nonetheless valuable as the first study in constructing a scalable, reliable, and dynamic fog infrastructure. Such work creates solid foundation for the development of future work in fog design & dimensioning, including the proposed PoC.

1.8 Organization

This is an article-based doctoral thesis. Each of the next five chapters presents an article that has been published, or submitted for publication. Chapter 2 present a survey on fog design, resource management and system evaluation tools [8]. Chapter 3 presents a scalable fog design &

dimensioning for a static fog infrastructure [39]. Chapter 4 presents a robust extension to the fog infrastructure [40]. Chapther 5 presents a fog-enabled UAV set-covering for 'bursty' IoT traffic [41]. Chapter 6 presents a blockchain-based system for enforcing fog integrity [42]. Finally, Chapter 7 concludes the thesis, and discusses previous and future work.

1.9 List of Publications

1. Martinez, I., Hafid, A. S., & Jarray, A. (2020). *Design, resource management, and evaluation of fog computing systems: a survey.* IEEE Internet of Things Journal, 8(4), 2494-2516.

2. Martinez, I., Jarray, A., & Hafid, A. S. (2020). *Scalable design and dimensioning of fog-computing infrastructure to support latency-sensitive IoT applications.* IEEE Internet of Things Journal, 7(6), 5504-5520.

3. Martinez, I., Hafid, A. S., & Gendreau, M. (2022). *Robust and Fault-Tolerant Fog Design and Dimensioning for Reliable Operation.* IEEE Internet of Things Journal, 9(19), 18280-18292.

4. Martinez, I., Hafid, A. S., & Gendreau, M. (2024). *Design and Dimensioning of a UAV Set Covering in High-Traffic IoT-Fog Environments.* IEEE Internet of Things Journal, **Submitted**.

5. Martinez, I., Hafid, A. S., & Gendreau, M. (2024). *A Blockchain-Based Audit Mechanism for Trust and Integrity in IoT-Fog Environments.* IEEE Transactions on Industrial Informatics, **Submitted**.

Chapter 2

Design, resource management, and evaluation of fog computing systems: a survey

Abstract A steady increase in Internet of Things (IoT) applications needing large-scale computation and long-term storage has lead to an over-reliance on Cloud computing. The resulting network congestion in Cloud, coupled with the distance of Cloud data centres from IoT, contribute to unreliable end-to-end response delay. Fog computing has been introduced as an alternative to cloud, providing low-latency service by bringing processing and storage resources to the network edge. In this survey, we sequentially present the phases required in the implementation and realization of practical fog computing systems: (1) design & dimensioning of a fog infrastructure, (2) fog resource provisioning for IoT application use and IoT resource allocation to fog, (3) installation of fog frameworks for fog resource management, and (4) evaluation of fog infrastructure through simulation & emulation. Our focus is determining the implementation aspects required to build a practical large scale fog computing infrastructure to support the general IoT landscape.

This chapter reviews the current state-of-the-art of fog-computing, which motivates the remainder of the thesis. This chapter has been published in IEEE Internet of Things [8].

Design, Resource Management and Evaluation of Fog Computing Systems: A Survey

Ismael Martinez, Abdelhakim Senhaji Hafid, and Abdallah Jarray

***Abstract*—A steady increase in Internet of Things (IoT) applications needing large-scale computation and long-term storage has lead to an over-reliance on Cloud computing. The resulting network congestion in Cloud, coupled with the distance of Cloud data centres from IoT, contribute to unreliable end-to-end response delay. Fog computing has been introduced as an alternative to cloud, providing low-latency service by bringing processing and storage resources to the network edge. In this survey, we sequentially present the phases required in the implementation and realization of practical fog computing systems: (1) design & dimensioning of a fog infrastructure, (2) fog resource provisioning for IoT application use and IoT resource allocation to fog, (3) installation of fog frameworks for fog resource management, and (4) evaluation of fog infrastructure through simulation & emulation. Our focus is determining the implementation aspects required to build a practical large scale fog computing infrastructure to support the general IoT landscape.**

Index terms— Fog computing, fog design & dimensioning, fog resource management, fog infrastructure evaluation, simulation, Internet of Things (IoT), survey

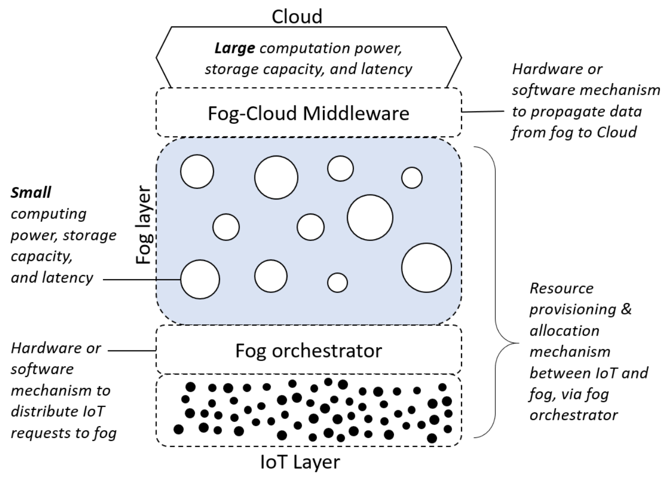

Cloud
Large computation power, storage capacity, and latency
Hardware or software mechanism to propagate data from fog to Cloud
Fog-Cloud Middleware
Fog layer
Small computing power, storage capacity, and latency
Resource provisioning & allocation mechanism between IoT and fog, via fog orchestrator
Hardware or software mechanism to distribute IoT requests to fog
Fog orchestrator
IoT Layer

Figure 1: High-level necessary features and components for a complete fog system. Larger nodes have more resources.

I. Introduction

The benefits and varied use cases of Internet of Things (IoT) technology have led to an increase in IoT adoption, number of devices and applications, and volume of data uploaded to Cloud systems. International Data Corporation (IDC) predicts the number of connected IoT devices will exceed 41 billion by the year 2025, generating more than 79 zettabytes of data [1]. Current Cloud systems are not large enough to process and store this increase in IoT data traffic [2], an issue that affects all IoT systems. Congested networks towards a distant Cloud can result in relatively large delay for latency sensitive IoT applications such as health care [3], multimedia [4], and vehicular/drone applications [5, 6]. Furthermore, Cloud centralization can result in reduced privacy of uploaded IoT data [7].

Fog is a layer of geo-distributed servers with computing, memory, and network capabilities that serve as an intermediary between IoT and Cloud layers. Compared to Cloud, fog servers sit closer to IoT devices, providing reduced response time latency able to service most latency sensitive IoT applications [8]. Although fog servers are much smaller in terms of processing and storage capabilities than Cloud [9], the larger number and geo-distribution of fog servers allow fog to alleviate Cloud network congestion by servicing a large load of IoT applications [8]. Indeed, an IoT application can be fully serviced by local fog servers without propagation of IoT data to fog or Cloud further into the network.

Despite a substantial amount of research proposals in fog computing, there are very few documented implementations of fog in large-scale environments [10]. A design and implementation of a fog system requires several components of the fog layer, as well as collaboration mechanisms with IoT and Cloud. A detailed overview of these components and collaboration mechanism are presented in [11, 12, 13], and are summarized in Fig. 1.

We identify three broad stages in implementing and developing a fog system. First, in the absence of any fog system, a fog infrastructure is built through consideration of IoT service needs. Since large IoT traffic volume may cause increased network congestion towards Cloud, building a localized fog infrastructure close to high traffic areas is beneficial to internet service providers and Cloud service providers alike.

Second, the approach the fog infrastructure interacts with IoT and manages fog resources is defined. Fog resource management aims to select fog nodes to best process IoT data, and takes the form of algorithms or protocols implemented within individual fog nodes or a fog layer controller. In several cases, resource management relies on additional hardware/software structures (e.g. fog orchestration controllers

I. Martinez is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ismael.martinez@umontreal.ca).

A. S. Hafid is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ahafid@iro.umontreal.ca).

A. Jarray is with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: ajarray@uottawa. ca).

or APIs) implemented within the fog infrastructure for appropriate fog node selection. Therefore, we consider the resource management algorithms/protocols, and the additional hardware/software structures separately. A *fog service provider* (FSP) is the organization that manages the implemented fog infrastructure, including IoT-fog interactions. Several proposed resource management approaches minimize service cost and latency to IoT users, which incentivizes IoT use of fog. Other approaches increase energy efficiency to reduce FSP operation costs. Therefore, efficient resource management for improved IoT service is an important concern to FSPs.

Third, evaluation of the constructed fog infrastructure with defined resource management protocols are evaluated to assess service impact on different IoT applications. Evaluation tools can be useful for IoT application developers to decide whether application computation is best done locally, by fog, or by Cloud. Tools for evaluation fog infrastructure use discrete-event simulation or emulation, and allow for configurable network conditions and IoT traffic patterns. To implement a fog computing system that can successfully support an IoT landscape, these three broad stages are focused into four key phases which will be explored in subsequent sections.

Phase 1: Estimate the volume of IoT traffic to be supported by fog, then **design & dimension** a fog infrastructure either from scratch, or by extending existing infrastructure.

Phase 2: Determine the method of fog **resource provisioning** for IoT use, and of IoT **resource allocation** to fog.

Phase 3: If necessary for resource provisioning, allocation and data migration, install a **fog framework** — additional hardware and/or software for fog resource management.

Phase 4: Use **fog evaluation** tools to measure the efficiency of the designed fog infrastructure and selected resource management approach in servicing IoT.

In this survey, we present a critical evaluation of solutions that contribute to the practical end-to-end implementation of fog computing. Our main contributions are as follows:

- Present and discuss existing models for fog design & dimensioning.
- Present a structured classification of resource management schemes based on initialization time and effectiveness for dynamic and static IoT applications. We also summarize optimization objectives and modeling techniques used.
- Review current framework solutions for fog resource management, and data migration, including an analysis of hardware and software overhead that is generated.
- Identify limitations of current simulation/emulation tools for the evaluation of fog infrastructures.
- Present open issues and research opportunities in practical fog implementation and evaluation.

Our discovery and selection of presented publications is meant to give a broad understanding of various proposed implementations of fog. We did select publications that cover the different facets of fog implementation. Indeed, publications are selected if they cover the following topics: (1) fog design; (2) fog resource management/orchestration; (3) fog evaluation, simulation or emulation; (4) fog applications; and (5) fog architecture. They are then categorized in the context of four phases. In this manner, we present a wide range of perspectives and approaches to fog implementation.

The remainder of this paper is organized as follows. Section II provides an overview of fog including motivation, architectures, and large-scale applications. Section III summarizes the limitations of current surveys in fog computing. Section IV reviews and compares current research in design & dimensioning of fog infrastructures. Section V categorizes resource provisioning & allocation schemes based on initialization time and effectiveness for dynamic or static IoT applications. Section VI surveys current proposals of fog frameworks for resource management and data migration across all fog nodes, and associated overhead. Section VII presents different simulation/emulation tools for the evaluation of fog infrastructures. In Section VIII, we discuss our findings, delineate the lessons learned and feature research challenges and opportunities for fog systems. Finally, Section IX concludes this paper.

II. Overview of fog Computing

We present an overview of the differences and benefits of fog to IoT when compared to Cloud and other edge technologies. Within the fog computing paradigm, we present several fog architectures that have been proposed to provide different ranges of IoT serviceability and inter-fog communication. Benefits and advantages unique to fog are particularly important to certain industries; hence, several industry applications of fog have been proposed for both local and large-scale implementations. Localization and geo-distribution of fog can provide increased privacy, but pose other security issues.

A. Definition of fog

Fog is a highly virtualized network comprised of nodes that provide processing and storage services to end devices (IoT) [8]. Fog nodes at the network edge, i.e. close to IoT devices, can provide computational support to IoT applications with minimal latency. Though typically at the network edge, fog nodes may appear hierarchically anywhere between the IoT layer and the distant Cloud [14]. This architectural setup allows many IoT requests to be satisfied by fog, and reduces the data volume reaching Cloud servers [15]. Since Cloud is still required by IoT applications that require high computation and long-term storage [16], the focus of fog is to support low resource and latency-sensitive IoT applications. According to [8], the main characteristics that define fog as a non-trivial extension of Cloud are: a) low latency and location awareness, b) wide-spread geographical distribution, c) support for mobility, d) very large number of nodes, e) predominant role of wireless access, f) heterogeneity, g) real-time interactions, h) interoperability and federation, and i) support for online analytic and interplay with Cloud. As a result, fog nodes can provide location awareness, activity awareness, time awareness, and energy awareness of IoT data processing [17].

Each fog node can be described as a small server or "micro data centre" [14] with available resources to support local computation. Although a typical fog node has reduced resource power compared to Cloud [9], the number of nodes in fog is

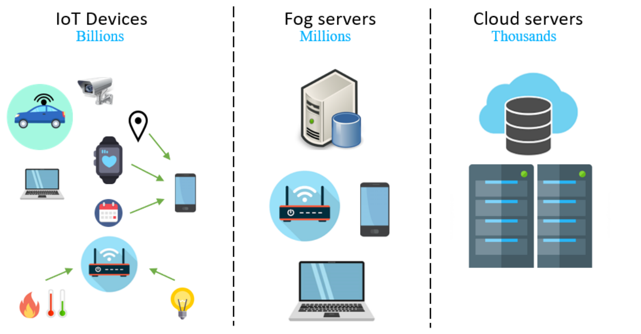


Figure 2: Examples and quantities of network devices per network service layer. IoT sensors may send data to a primary IoT device before data is transmitted and processed by fog/Cloud.

much larger [8]. The types and quantities of devices per system layer are described in Fig. 2. We summarize the difference between fog and Cloud in Table I. For the remainder of this paper, the terms “fog node” and “fog server” are used interchangeably.

Routing of data through fog often follows a path computing process [18] in which data is propagated to nodes of increasing size towards Cloud. Therefore, IoT, fog and Cloud layers can be expressed in a hierarchical architecture as in Fig. 3. Physically, fog servers are geo-distributed to be closer to IoT, providing the network structure in Fig. 4. Since each fog server has network connectivity, an IoT application may access any fog node either directly, or through a network access point.

A single physical IoT device may run multiple IoT *applications*. An IoT application that requires external computation or storage submits an IoT *request* to fog or Cloud. Each IoT request can be processed by one or more fog servers by partitioning into multiple *tasks*. Allowing for a request to be partitioned over several fog servers can result in parallelized execution of tasks, and decreased end-to-end response latency. We define a *static* IoT application as running on a fixed-location IoT device with frequent requests; otherwise, it is a *dynamic* IoT application.

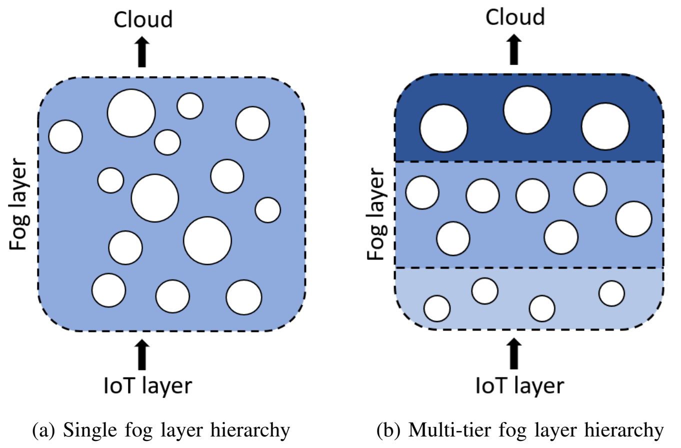

Cloud
Fog layer
IoT layer
(a) Single fog layer hierarchy
Cloud
Fog layer
IoT layer
(b) Multi-tier fog layer hierarchy

Figure 3: IoT data is often transmitted to fog prior to being transferred to Cloud, therefore the IoT-fog-Cloud system is often viewed as a hierarchy. Within the fog layer, many routing protocols restrict data propagation to larger fog nodes (i.e. more resources), creating a multi-tier fog hierarchy.

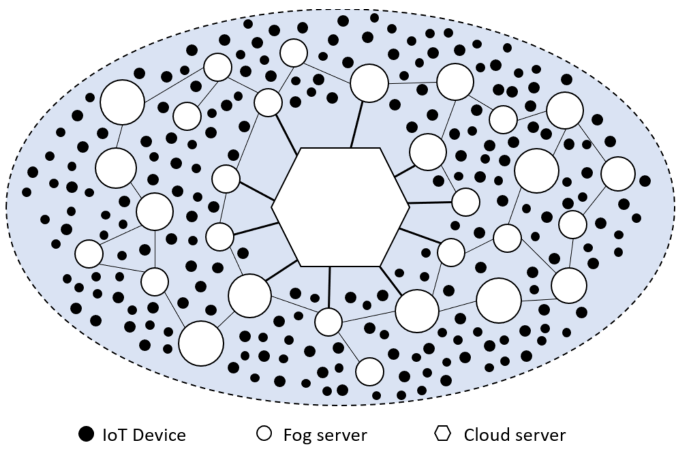


Figure 4: IoT-fog-Cloud physical network architecture. Larger nodes have more resources. Network links may be wired or wireless, and may involve intermediate routers/switches.

B. Comparison of fog with other edge technology

Edge computing was introduced to bring storage and processing capabilities closer to IoT users for localized and low-latency computation [19, 20, 21]. Though the greatest benefits of edge computing occur when edge servers are in close proximity of data sources, edge computing is defined as being any computing and network resource between IoT users and cloud [19, 20]. Therefore, fog is seen as one implementation of edge computing [19, 20, 21]. Two other well-known implementations are *cloudlet computing* and *mobile edge computing* (MEC) [21].

A *cloudlet* is a small cluster of computers, forming a “data centre in a box” [22], and may be referred to as a *micro-cloud* [23] in some literature. Cloudlet computing refers to any implementation of cloudlets in the vicinity of end users. Cloudlets provide one-hop, high-bandwidth and low-latency wireless access to IoT users.

MEC, also known as *mobile cloud computing* [24], is a form of edge computing implemented within the Radio Access Network, and is therefore specific to mobile devices. MEC nodes are co-hosted at Radio Network Controllers or base stations, such as cell towers [25]. As a result, MEC nodes are always one-hop communication distance from active mobile users, and provide real-time processing of mobile requests. However, MEC does not provide edge services beyond direct network connectivity [25].

Fog nodes may be dedicated devices, but may also be legacy devices augmented with storage and computation resources [21]. This allows the fog infrastructure to be flexible since any device may become a fog node. However, fog nodes may be several hops from IoT users, and may require cooperation between multiple fog nodes for IoT processing. Although cloudlets and MEC nodes have no intercommunication, they are dedicated devices with high resource capabilities and direct access to edge users. Hence, a cloudlet or MEC node is meant to process all IoT requests in full without further propagation. Cloudlets are depicted as being located within hotspot areas, such as hospitals or educational institutions [26], and MEC nodes are co-hosted with base stations [25]. Compared to fog nodes, MEC nodes and cloudlets may provide lower latency due to higher resource capabilities and one-hop proximity, but have lower implementation flexibility [21].

Across fog computing, cloudlet computing, and MEC, various service providers may own and manage a small subset of available devices. Without standardization of connectivity software, local IoT users may be limited to using a subset of nearby edge devices. When considering a large scale implementation towards Smart City technology, these restrictions can become a road-block for a fully connected digital ecosystem [27]. It is therefore encouraged for all edge computing to implement network protocols and software interfaces that provide unified IoT connectivity.

Our focus for the remainder of the paper will be largely on fog – i.e. systems where there exists intercommunication among nodes. We will use cloudlet and MEC when appropriate.

C. Fog Architecture

Many fog architectures in literature use a single layer stretching from IoT to Cloud layer, allowing any two fog devices to share data [3, 28, 29, 30]. This is the most general and flexible representation of the fog layer, with each fog node varying in distance from the edge and quantity of resources.

Intharawijitr et al. [31] propose a layer of horizontally placed fog nodes that cannot communicate with sibling nodes; instead, each fog node is restricted to communicate with only IoT and Cloud layers. In practice, processing latency can be reduced by having IoT upload data to the nearest fog node, and permitting fog nodes to migrate data amongst themselves if more resources are needed [28, 29].

Some fog architectures are represented as multi-tier hierarchies, with data sharing available across different fog layers but not within the same layer. Fog nodes are divided by computation power, memory, storage capacity, and proximity to IoT devices. IoT devices upload data to the first fog layer, which then uploads to higher layers until a fog node is found with sufficient resources. This architecture has been represented with two layers [32, 33] and three layers [34, 35]. iFogSim [36] defines a structure of multiple fog layers based on distance from Cloud, while [37] represents fog as a tree of fog nodes rooted by Cloud. Instead of tiers/layers of fog nodes, [29] and [38] define clusters of fog devices, with intra-fog and inter-fog communications. Similarly, [39] partitions fog nodes into clusters and has a hierarchy within partitions of the nodes.

Tang et al. [33] proposed a hierarchical architecture with Smart Cities in mind; the first layer sits at the network edge, the second layer is composed of larger fog nodes covering neighbourhoods, and the third layer uses largest fog nodes connected to Cloud to support city services. Arkian et al. [17] develop crowd-sensing applications supported by IoT and fog for small scale city services such as parks. Sun and Ansari [40] propose a hierarchical extension to MEC that connects to fog. Each cellular base station is connected to the fog infrastructure to alleviate edge traffic and handle large mobile data streams.

D. Motivation

The large incentive of using fog computing is its ability to process IoT data with real-time or latency-sensitive requirements. Use cases that benefit from fog cover health care, autonomous vehicles, and multimedia.

1) Health Care: Gill et al. [41] propose to use body sensors with fog to help diagnose heart disease. It has also been proposed to use fog with wearables and sensors [3, 28] to provide real-time assisted living services to patients in hospitals or health care centres.

2) Autonomous Vehicles: Loke et al. [6] proposed an asset management concept for autonomous drone technology. A fog server can provide control signals for drone navigation, given a line-of-sight between a drone and the fog server. Fog servers can also relay traffic condition information from smart vehicles [5, 29], creating area-wide, real-time traffic sharing; this ultimately reduces road accidents. Coupled with smart traffic lights [42], fog increases the efficiency of route navigation.

3) Multimedia: Due to the per-instance and real-time processing fog can provide, fog has been proposed to deliver processing to multimedia, such as gaming, video streaming, and augmented reality [4]. Video surveillance applications can use fog for facial recognition, diminishing the response time of appropriate authorities in the event of an incident [37, 43]. Surveillance cameras at the scene of an incident can create bursty data; a decentralized fog infrastructure can process different data in different fog nodes, resulting in an overall quicker emergency response [43].

For IoT applications in health care, smart vehicles and multimedia, a difference of milliseconds in response time can lead to a significant impact on event outcome. In these cases, fog computing can provide overwhelming benefits in supporting IoT applications at the network edge.

E. Industry Applications

Applications of fog computing systems have been proposed in several industries to take advantage of the unique traits of fog. Applications range from local to large-scale implementations.

1) Internet of Vehicles: Fog and road-side units (RSUs) can collect traffic information from smart vehicles. Integrating fog with an Internet of Vehicles (IoV) infrastructure can allow cars to participate in sharing real-time traffic conditions throughout the city [5, 29]. Other fog applications of IoV infrastructure use clusters of slow moving or parked cars as the fog itself [44, 45]. Sookhak et al. [45] propose incentives for participating such as free parking, free Wi-Fi or free shopping vouchers, while Hou et al. [44] show how non-smart cars may be upgraded with hardware and/or software in order to take part. For fairness and participation motivation, it is proposed that incentives for participating vehicles correspond to the quantity and type of contributed resources [46, 47], and provide privacy & security to vehicles and users [46].

2) Health Care: Santos et al. [3] argue the use of fog for e-health monitoring systems by providing a stochastic analysis of fog server reliability when backed by Cloud. They conclude that fog-Cloud system failures are small enough (under 0.3%) that they do not nullify the benefits of fog. Ahmad et al. [48] consider the use of fog to provide better control of data privacy & security of smart phone health applications.

Gateways act as intermediaries between sensor networks and Cloud systems; Rahmani et al. [49] envision the use of fog-enabled *smart e-Health Gateways* to support local computation and storage for body-worn or implanted sensors in a smart hospital or home. Medical cyber-physical systems (MCPS) provide seamless connection between healthcare devices and computational resources. Gu et al. [28] introduce a fog infrastructure to support MCPS.

3) User Provided Fog: More generally, *Consumer as a Provider* platforms allow for user devices such as phones and modems to act as fog devices and are made available to the public [50, 51]. With a large enough user base, this fog infrastructure can become very large in scale covering a wide area such as a city.

4) Smart Cities: The concept of a Smart City is to improve the life of citizens through integrated monitoring and adaption of city services [52]. Examples include Smart Grid, Smart Transportation, and crowd-sensing applications.

Smart Grids are electricity networks updated with smart meters and shared customer usage information to service providers. This incoming information determines how much electricity should be generated and where it should be sent. Okay and Ozdemir [53] propose the use of fog computing for scalable real-time electrical usage monitoring and improved privacy of information sharing.

Smart Transportation applications attach IoT sensors to public transportation such as buses and subway trains to share real-time information on transport location and delays [54]. Road-side and fog computing infrastructures can facilitate optimal route calculations and collision avoidance for smart cars, including self-driving cars, via real time traffic conditions over connected vehicles [5, 29]. Smart traffic lights optimize traffic by flow using vehicular sensors and traffic cameras [55].

Bittencourt et al. [56] propose a fog architecture that provides real-time IoT application allocation and processing, especially suited for mobile IoT such as smart phones and smart cars within a city. Non-invasive and low-cost static sensors can be set up in densely populated public areas to provide real-time crowd-sensing services when used alongside fog [17]. Installed in an outdoor park setting, crowd-sensing devices can passively determine in which areas the most activity is taking place, resulting in awareness of possible maintenance or updated amenities. This concept can be extended to other Smart City use cases such as monitoring air pollution or noise pollution from mounted sensors on outdoor and indoor public transportation respectively.

Unmanned Aerial Vehicles (UAVs) can be quite useful in providing additional computation services to IoT in a Smart City environment. One approach is to dispatch UAVs over environments of large IoT traffic to provide direct IoT support [57, 58]. Since these UAVs exhibit cloudlet and MEC functionality, they are designed to process the full amount of IoT resource requirements without propagation, which may create large resource and energy demands on UAVs. A more robust fog-enabled approach is to dispatch a UAV over low service areas and connect to the surrounding IoT and fog infrastructure [59]. Low service areas may be a result of abnormally high IoT traffic, a failed fog node, or areas that are challenging for human or manned vehicle access.

More city focused use cases include Smart Agriculture, Smart Health & Well-being, Smart Waste Management, Smart Water Management, Smart Greenhouse Gas Control, Smart Retail Automation [60], Smart Pipeline Monitoring [33], Noise Pollution Mapping, Urban Drainage Networks [10], augmented reality [15], City Structure Health Monitoring, Environmental Monitoring, and Public Safety & Security [55]. Together, they become components of a sustainable Smart City supported by fog for real-time information queries [55, 60].

F. Privacy & Security

Fog provides computation and storage resources over servers that are geographically distributed, providing a means to isolate IoT data computation and/or storage to localized fog servers. This layout allows sensitive IoT data, such as health care, to never leave the vicinity, keeping the data from being collected and used by unwanted parties such as tech giants [7].

The distribution of fog nodes and servicing of heterogeneous IoT can lead to security issues between nodes. Dzousa et al. [61] propose a policy-based management framework to support secure communication, collaboration, and interoperability of requested resources in fog. Liu et al. [62] use hash puzzles distributed to nearby vehicles to eliminate possible denial-of-service attacks to smart traffic light systems with fog capabilities.

Network security and congestion can pose a problem to fog in providing low-latency services. The CloudWatcher framework [63] uses OpenFlow [64] to monitor the network for intrusion detection and other security risks.

The majority of research into privacy-preserving communication and data security uses homomorphic encryption from IoT, or attribute-based encryption between an IoT-fog pair [65]. Since the focus of this survey is in the implementation of fog regardless of IoT behaviour, we do not further discuss privacy & security issues in fog beyond an awareness of their existence.

III. Existing Surveys on fog Computing

Hu et al. [66] explore the characteristics and benefits of fog when used with IoT and Cloud. They present a comparison between Cloud computing and fog computing paradigms. They also present an in-depth description of computation, storage, and communication technologies used in fog. Dolui et al. [21] discuss the concepts, benefits and technologies of edge computing. They provide a detailed comparison of the three main paradigms of edge computing: fog computing, cloudlet computing, and mobile edge computing (MEC).

Table I: Differences between fog and Cloud layers.

Feature	Fog	Cloud
Latency	Low	High
Distribution	Geographically distributed	Centralized
Distance to network edge	Close	Far
Number of nodes	Millions	Thousands
Resource size	Small	Large
Access	Predominantly wireless	Wired and wireless
Heterogeneity	High	Low
Interaction with IoT	Real-time	Batch processing
Owned & Managed	Various service providers	Few large organizations

Mukherjee et al. [67] study advancements and benefits derived from integrating fog into current technologies, such as virtualized fog data centres, fog radio access networks, and software-defined network (SDN) enabled fog architectures. Resource allocation models and techniques are discussed alongside mathematical models of fog components such as latency, energy consumption, and resource sharing.

Mouradian et al. [68] present a comprehensive review of major contributions in fog covering six criteria of heterogeneity, QoS management, scalability, mobility, federation, and interoperability. Ghobaei-Arani et al. [69] provide a systematic and comprehensive literature review of resource management issues and solutions in fog computing. They classify mechanisms and techniques into application placement, resource scheduling, task offloading, load balancing, resource allocation, and resource provisioning. Brogi et al. [70] present an exhaustive overview of resource allocation solutions within fog. Surveyed contributions are further classified based on an algorithmic perspective which looks at solution methodology, and a modeling perspective which looks at constraints and optimization metrics. Neither [68, 69, 70] consider the time overhead of resource provisioning & allocation prior to IoT processing. Indeed, though the assigned fog servers to an IoT application may provide optimal latency, the assignment process may be too slow for time-sensitive IoT applications.

Naha et al. [71] review the publication trends of fog computing and Cloud computing alike, and present a taxonomy of fog research publications by requirements of infrastructure, platform and application. They provide an overview of other technological architectures analogous to fog such as edge computing and dew computing. Mahmud et al. [11] identify key challenges and properties of fog computing, and use them to provide a taxonomy of aspects in fog computing such as fog node configuration, nodal collaboration, service level objectives, applicable networking system and security concerns. Ahmed et al. [12] select and review 30 actual or proposed fog applications. The selected contributions have little overlap, and cover a broad range of industries, uses and communication methodologies. Selected reference applications are used to study reasons for using fog, required fog hardware platforms, assumed data distribution methods among fog, leveraged fog service models, privacy & security requirements, and application workload characteristics on fog. Yi et al. [72] describe the issues potentially faced when designing and implementing a fog system, such as in IoT communication interface, computation offloading, accounting, and resource management. Yi et al. [65] describe the privacy & security issues that arise from IoT-fog communication. Geo-distributed and edge location features of fog can expose an IoT device's location to a small radius around the connected fog node, in addition to possible exposure of application data and usage frequency to fog. From the IoT perspective, the owner of a fog node is not always evident, resulting in trust and security issues when connecting to an arbitrary and close fog server.

Markus and Kertesz [73] provide a taxonomy of simulation tools and environments for fog and edge computing. They propose a taxonomy of available simulators modelling fog, edge, Cloud and IoT networks to aid researchers in distinguishing the right tool for different research needs.

Across these surveys, open issues and research challenges in fog computing are discussed [11, 12, 66, 67, 71, 72], applications of fog computing to IoT use cases are summarized [12, 66, 67, 68, 71], and gaps in current research towards future work are identified [66, 71, 72]. Some survey review individual components of the fog computing system such as resource provisioning & allocation and fog frameworks [68, 69, 70], security & privacy [11], and fog simulation software [73]. All discussed open issues address algorithmic enhancements to resource management techniques of existing fog infrastructures. Additional hardware may also mitigate these issues at the cost of generated overhead. However, changes and additions to fog design are not covered. Our survey, on the other hand, provides a holistic view of the applicability, challenges, overhead, and limitations of proposed fog systems, irrespective of fog technology used.

To conclude, we summarize the limitations of existing surveys as follows: (1) none covers the four phases to realize fog systems; (2) none covers design & dimensioning of fog systems; (3) none analyzes the generated overhead of framework implementations; (4) none covers the multiple migration scenarios between fog servers; (5) none considers resource provisioning allocation time overhead prior to IoT request processing [68, 69, 70]; and (6) none considers changes to fog design to mitigate open issues.

In this paper, we intend to detail the full fog implementation process, beginning with an IoT environment without any available fog system. We review and compare the current contributions for designing & dimensioning a fog infrastructure. We identify efficiency, assumptions, shortcoming, and generated overhead of resource provisioning & allocation schemes, and fog frameworks. We review the features and efficiency of simulation/emulation tools for the evaluation of a designed fog infrastructure, implemented resource provisioning & allocation mechanism, and conceptualized fog framework. Finally, we identify the limitations and open issues of all components of fog implementation.

IV. Fog Design & Dimensioning

For users and organizations wishing to implement their own fog infrastructure to support local edge devices, Mahmud et al. [11] outline the ground work for what components and mechanisms are necessary in fog (see Fig. 1). This does not however give insight into the location and quantity of installed resources of each fog node, known as *design* and *dimensioning* respectively.

Design of edge networks to provide low-latency access and processing to IoT has been studied with cloudlet computing, which involves no intercommunication between cloudlets [74, 75, 76, 77]. In most cases, cloudlets are designed to be one-hop away from IoT devices; however Ceselli et al. [76] propose an augmentation to MEC where data is routed from base stations to a nearby cloudlet. Though other contributions are only interested in network placement of cloudlets, Fan and Ansari [77] dimension the number of cloudlet servers installed in each designed cloudlet location. Building a dedicated computing infrastructure with high resources may be costly, and underused in most cases. A fog system provides more network flexibility and geo-distribution of smaller devices, potentially covering and servicing a much wider IoT ecosystem. To our knowledge, there are currently only two contributions that develop an optimal fog design scheme. Both schemes optimize fog node locations and installed computing & memory resources while satisfying IoT QoS requirements.

Yu et al. [5] consider fog to provide real time processing for autonomous vehicles. They propose a fog design & dimensioning scheme of RSUs, fog nodes, and Internet Gateways which work together to provide real-time traffic information to enhance and facilitate automated navigation and collision-avoidance. Given a set of candidate locations, candidate resource configurations, and a known number of connecting vehicles for a certain location and time period, the location and resource amounts of RSUs, fog nodes and Internet Gateways are optimally found via a Mixed Integer Linear Program (MILP) in an arrangement that minimizes infrastructure costs. RSUs may be fog nodes themselves (coupled variant) or are separate from fog nodes (decoupled variant); both variants are tested and compared. They conclude that a decoupled model allows design flexibilities that result in a more economical and cost-effective scheme. For scalability, a heuristic algorithm based on the decoupled model is used.

Regarding vehicular traffic, the model assumes a known static set of vehicle resources accessing the network across different regions, which may not be true in practice. Although having a set of candidate locations for fog nodes is practical, the model also assumes a finite candidate set of dimensioning configurations. The solution to this model is thus dependent on the completeness of such a set. Finding the optimal placement of RSUs and Gateways increase the complexity of the model, while the inclusion of RSUs also restricts the application of this model to IoV.

Martinez et al. [78] propose a fog design & dimensioning scheme to support the general IoT landscape. For a given area, the future IoT data volume and the resulting stochastic network congestion distributions are estimated, which affect the approximated IoT-fog end-to-end communication delay. An IoT request with k tasks is represented by a Task Dependency Graph of k nodes, and the set of physical candidate fog node locations are represented by a bidirectional graph. Tasks and task dependency links are mapped to physical fog locations and fog infrastructure paths respectively. A fog design & dimensioning scheme is defined to find a mapping that satisfies fog node resource capacities, fog infrastructure bandwidth capacities, and IoT QoS requirements.

An MILP model (fog-DC-MILP) is used to find an exact optimal fog infrastructure by minimizing infrastructure deployment costs. Due to the intractability of the fog-DC-MILP model, a Column Generation model (fog-DC-CG) is proposed.

Simulation and scalability testing between fog-DC-MILP and fog-DC-CG show a significant reduction in solution computation time of fog-DC-CG with near-optimal cost. Furthermore, fog design & dimensioning solutions of both models are similar. This indicates fog-DC-CG is a practical alternative model to fog-DC-MILP.

For each candidate fog node, they define a maximum amount of each resource that can be installed, allowing for resource configurations selected from a continuous set. This aims to remove the concern of discrete resource configuration set completeness observed [5]. The designed & dimensioned infrastructure [78] is extensible, allowing for current fog nodes to be upgraded or extra fog nodes to be added to the current infrastructure if IoT data volume increases.

They use a discrete set of IoT devices, each uploading data at a rate following a Poisson Process. They compute a percentile estimation of expected IoT traffic and resulting congestion to produce a deterministic upper bound. This allows for simple modifications to beginning percentile parameters to increase or decrease traffic estimation, resulting in a change in the fog design & dimensioning solution.

Due to the models extensibility, it is reasonable to make an underestimate of expected IoT traffic and extend the designed & dimensioned infrastructure based on future performance. These two contributions [5, 78] are summarized and compared in Table II.

V. Fog Resource Provisioning & IoT Resource Allocation

Fog *resource provisioning* refers to the reservation of computational and memory resources within fog nodes for use by IoT applications. IoT *resource allocation* refers to the assignment of resource requirements for an IoT request to the fog. It is clear that resource provisioning and resource allocation are two sides of the same coin, since a fog node needs to provision its resources for the allocation of IoT requests. For all resource provisioning & allocation schemes, several consistent infrastructural assumptions include: 1) existence of a pre-defined fog infrastructure, 2) all fog nodes are reachable and available from any IoT device, 3) any pair of IoT device and fog node experiences static network congestion and/or latency, and 4) uploaded data format and response is homogeneous.

Table II: Description of contributions in fog design & dimensioning.

Scope	Yu et al. [5]	Martinez et al. [78]
Main Contribution	Scalable design & dimensioning for fog nodes, RSUs and Gateways to support autonomous vehicles.	Scalable fog design & dimensioning to support general IoT systems with near-optimal implementation cost.
Supported IoT devices	Smart Vehicles.	General IoT devices.
Pre-defined device candidates	Fog nodes, RSUs, Gateways.	Fog nodes.
Predicted IoT traffic	Discrete vehicular traffic set.	Percentile of stochastic IoT traffic predictions.
Resource dimensions	Selected from discrete pre-defined set.	Continuous up to a maximum capacity.
Extensible	—	Fog nodes can be added to current fog infrastructure to account for increased IoT traffic.
Congestion	—	Accounts for possible network congestion by designing fog under worst-case network scenarios.

For IoT applications that intend extended use by the same fog servers, the server partitions a *module* of reserved resources. Once a module is set for an IoT application, all requests from that application are immediately processed by the module for reduced long-term latency. However, the reservation of fog resources itself may take time, and may not be useful for IoT applications which require immediate and infrequent processing. Module *migration* is the process of freeing module resources in the current fog node, and re-provisioning the module in a different fog node or Cloud. Any IoT processing data or storage present in the current fog node is transferred to the new module.

The allocation schemes [9, 31] assume each IoT is comprised of a single task, while Agarwal et al. [30] propose to process each request by a single fog node, split into multiple tasks if there are insufficient resources. In both cases, the entirety of the IoT data need only be processed by a single fog node. Instead of mapping each IoT request separately, Yousefpour et al. [43] go further by clustering IoT devices together that run the same service, and map those services to fog node modules. Since all computation of an IoT application is done on a single fog server, many requests can result in a large processing queue and high latency.

Taneja et al. [79] report that each IoT request is in fact comprised of multiple tasks — stemming from multiple sensors and actuators — that are too taxing to be processed on a single fog node. Therefore, an IoT application's multiple tasks are split among one or more fog nodes. The multiple tasks of a single IoT request are often depicted as a directed acyclic graph, with each directed link representing a dependency between tasks [79, 80]. Hence, fog processing of tasks may require the same task workflow order. Similar approaches are to allocate application tasks to different fog nodes either via algorithms [37, 79, 80], policies [16, 36, 56] or optimization models [32].

In our review of current literature in this space, we did observe that all proposed models fell into one of three classifications based on prompt or optimal service to IoT applications. The fog layer may reserve resources for future IoT use based on accurate IoT traffic predictions. When IoT requests arrive, it is assumed the resources are available and immediate processing takes place. Schemes that follow this approach are known as *prior provisioning and prompt allocation* schemes.

Since IoT traffic predictions require additional computation effort and may be faulty, most schemes will provision resources only once IoT requests have arrived. There are two general approaches to resource allocation using on-demand provisioning: prompt allocation and small batch allocation. When an IoT allocation is promptly serviced, it is sent to the nearest fog node for processing, regardless of cost. Models that follow this approach are known as *on-demand provisioning and prompt allocation* schemes, and are ideal for dynamic IoT support. Another approach is for a fog resource manager to accumulate a small batch of IoT requests, and find the optimal allocation of IoT tasks that optimize some metric (e.g. latency, resource cost). Once a module is provisioned for an IoT request, all future requests from the same IoT application are provided immediate service by the module. Hence, small batch allocation provides efficient IoT processing for the life-span of the provisioned module. Models that follow this approach are known as *on-demand provisioning and small batch allocation* schemes. Though these schemes provide more efficient long-term IoT support, execution of these schemes are significantly slower than prompt allocation methods. Therefore, small batch allocation schemes are better suited for static IoT applications. A summary of these three classifications of resource provisioning & allocation schemes is provided in Table III and Fig. 5.

A. Prior provisioning and prompt allocation

Aazam and Huh [14, 81] propose the analysis of fog resource usage data of IoT devices to determine the relinquish probability a new service request will be abandoned within a time frame. IoT devices with low relinquish probability, i.e. will continue fog usage for long periods, are offered slightly lower usage prices as well as higher allocated resources by fog. Behavioural analysis of previously connected IoT ensures that enough resources are reserved by fog for future predicted IoT traffic, and allows for an immediate connection and processing of IoT data upon a new request. For IoT devices that have never connected prior, a default low relinquish probability is assumed, and resource pricing and quantity are calculated accordingly in real-time. This concept is extended in [82, 83]

Table III: Process of resource provisioning & allocation schemes by prompt or optimal service.

Resource Provisioning		Resource Allocation: Prompt	Resource Allocation: Small Batch
	Prior	• Reserves resources in fog based on historic predictions of future IoT traffic. • IoT requests are allocated immediately to reserved resources. [14], [81], [82], [83]	NA
	On-demand	• An IoT request arrives to a fog node; the fog node verifies if it has sufficient resources. • If yes, it processes the IoT request. Otherwise, it propagates request to a further node. [29], [43], [30], [47], [56], [84], [85]	• Groups several IoT requests for batch resource provisioning & allocation. • IoT requests are distributed in fog to optimize efficiency. [4], [9], [17], [28], [31], [32], [79], [80], [86], [87], [88], [89], [90], [91], [92]

to also consider historical quality of experience (QoE) of IoT based on end-to-end delay, jitter, packet loss, latency, and blocking probability. This approach more efficiently predicts future resource consumption for real-time allocation for multimedia [82] or haptic sensors [83].

Although an IoT application's QoS is satisfied, QoE may be low and thus requires different fog node connections or more fog resources on future requests. Instead of requiring the allocated fog node to satisfy the required latency of the IoT application, [14, 81, 82] focus on providing high fog utilization assuming any fog node could satisfy the IoT latency requirements. On the other hand, [83] ranks potential fog nodes for IoT allocation by latency, and verifies the latency suitability of the fog node before assigning resources.

B. On-demand provisioning and prompt allocation

Agarwal et al. [30] propose a resource provisioning scheme that does not rely on any current or historical information from IoT or fog. The proposed algorithm begins with an IoT application sending its request to an arbitrary fog node within communication range, usually the nearest node. If the fog node has enough resources to process the request, it will do so; otherwise, the request is partitioned into several tasks, and are sequentially processed by the limited fog resources. If no resources are available in that particular fog node, the IoT request is propagated to Cloud. In this worst case scenario, the propagation to Cloud may result in high latency and unsatisfied IoT QoS. Intermittent fog resource sharing among the fog could allow the initial fog node to know which other fog nodes have enough resources [93], and forward accordingly thus keeping latency low.

Bittencourt et al. [56] extend this idea by introducing three possible fog processing policies. In all cases, an IoT application is assigned to the first fog node with which it connects, but the processing order of IoT applications differs. If no resources are available at the fog node, Cloud propagation is applied. The *concurrent* strategy performs fog processing on IoT data regardless of current available resources; applications within a fog node are processed in parallel, and the allocation indifference to current resources could lead to high processing latency. The *First-Come-First-Serve* (FCFS) strategy processes IoT requests in the order of their arrival to a fog node. The *delay-priority* strategy processes the IoT application with the lowest QoS latency requirement first, and re-orders the next IoT application to process as new requests arrive. Simulations show that the FCFS and delay-priority strategies yield the lowest latency, whereas the concurrent strategy yields the lowest amount of data transferred to Cloud.

For vehicular fog settings where mobile vehicles require computation from static or slow-moving vehicles, Peng et al. [84] propose a multi-attribute double auction mechanism for base stations to match and pair vehicular fog nodes with vehicular IoT users. The mechanism allows vehicular fog nodes to announce their resource attributes, reputation and asking price, which is met with IoT announcements of resource and latency requirement, and bidding price. The formulated one-to-one assignment algorithm for resource matching executes in under 8 milliseconds for up to 100 vehicular fog nodes and up to 100 vehicular IoT devices. This scalable matching algorithm adds very little to the overall near instant computation provided by the allocated vehicular fog node. Similarly, Zhou et al. [47] introduce a contract-based mechanism for IoT request offloading to nearby smart vehicles. A contract is designed and offered to a vehicle based on the amount of resources and time in return for a reward; IoT users requiring fog computation are paired to a vehicular fog via a pricing-based stable matching algorithm.

Zhang et al. [29] introduce a cooperative fog computing architecture to deal with big IoV data. It allows for data migration between fog nodes for mobile smart cars. As a result, two separate resource allocation strategies are considered based on available resources at the nearest fog server. Each fog node in this system has a finite set of Virtual Machines (VMs) which partition fog resources for IoT use. If the number of VMs in a fog server is sufficient to process an incoming IoT application, intra-fog resource management will allocate the application to VMs that minimize fog energy-consumption via convex optimization. If data migration is required, a min-max optimization model is used to determine to which fog node data should be transferred to minimize the transfer rejection probability. The delay of data transfer between fog nodes is measured at 30 – 70 ms, providing low additional delay to the allocation process.

Yousefpour et al. [43] address the problem of dynamically deploying or releasing IoT services on fog nodes by means of two possible greedy algorithms that comply with QoS requirements. Both algorithms determine periodically from which fog node modules should be released and transferred to Cloud, in order to free resources for future IoT requests. The min-cost algorithm aims to allocate IoT services to the fog node that would provide the lowest resource allocation cost; it similarly releases IoT services that incur large resource costs. The min-viol algorithm allocates IoT services with high demand and releases services with low demand from fog nodes. The response latency inherently increase for modules released to Cloud, and thus results in a QoS violation.

Xia et al. [85] propose two ordering-based heuristics to search and select fog nodes to which an IoT application is allocated. With *anchor-based fog node ordering* (AFNO), fog nodes are ordered by its latency to an anchor IoT application. *Dynamic component ordering* (DCO) attempts to allocate IoT tasks until a constraint failure in the search occurs; the components are reordered with the failed tasks being allocated first, and may require several reorders for a successful application allocation. Ordering fog nodes by latency adds sorting overhead to AFNO, resulting in lower execution times with DCO allocation. For up to 20,000 fog nodes, DCO executes in under 100 ms for a single IoT application, allowing for real-time resource allocation and execution.

Reinforcement learning (RL) is a machine learning technique that aims to make improved decisions over time based on the reward or penalty incurred by previous actions [86]. In many RL approaches to resource allocation [87, 88, 94], this training process is divided into two parts: 1) receive IoT request, feed through RL model and take action, and 2) update RL model based on action reward/penalty. This process prioritizes the resource allocation decision, providing real-time support to IoT. Since RL techniques improve with time, the RL models are initialized with random values to start, meaning the initial resource allocation decisions may not be optimal. If the RL model is trained with batches of IoT requests, then the request data is saved in memory until enough requests are accumulated for training [94]. Since RL can be computationally taxing, it is often proposed to use a separate fog server to conduct all RL modeling [87, 88].

Sun et al. [94] use power consumption of fog nodes as a reward/penalty to model an energy-efficient resource allocation scheme. The action taken defines which fog processors to turn on or off, and to which layer to send an IoT request (fog or Cloud). Wei et al. [87] use a single fog server to compute all RL model updates, which receives and distributes IoT requests to the remaining fog servers. The formulated model aims to optimize response latency of resource allocation and content cashing.

Wang et al. [88] propose a RL model for fog allocation in IoV environments for minimizing response latency. All IoT requests are sent to an independent and centralized fog sever, where all RL model updates are computed. The fog server returns a decision to the mobile device of which layer to access: cellular network, device-to-device network, or fog network. Then, the mobile device sends another request to the appropritate network for processing. This procedure requires communication with two separate nodes which may increase latency. Furthermore, although QoS requirements of IoT, energy consumption of fog and total latency are considered, the resource capacities of each fog node are not. It is assumed that each layer can always process incoming IoT requests, and the main consideration is the best distribution to do so.

C. On-demand provisioning and small batch allocation

Zeng et al. [89] research fog supported Software Defined Embedded Systems with client-side computation support. In this scenario, client-side computation incurs a certain cost with client-side computation latency, whereas fog incurs a different resource cost, and both transmission and computation latency. For each IoT device, computation placement (client-side or fog) is formulated as a Mixed Integer Non-linear Program (MINLP) to minimize overall processing latency. This MINLP is linearized, and re-formulated as a three-stage heuristic. Although it may be convenient to assume an IoT device has client-side processing capabilities, this assumption is not practical for a general IoT ecosystem.

Souza et al. propose a resource provisioning scheme where fog node resources are partitioned into *slots* of fixed size, with IoT requests requiring a set number of slots. In particular, it is assumed that an IoT request either requires a small number of slots and can thus be processed anywhere in the fog, or requires a large number of slots and can thus only be processed by second-tier fog nodes. Formulated as an MILP, this resource provisioning simulation setup is wildly simplistic and requires more dynamic parameters to approximate a practical fog system scenario.

Zhang et al. [90] introduce *massive data centre operators* (MDCOs) as a middle-man between fog and IoT. A Stackelberg game is played between fog and MDCOs, and MDCOs and IoT devices to determine an optimal resource price and amount provided by fog and MDCO to maximize fog utilization.

Ali et al. [92] propose a many-to-one matching game between a set of fog nodes and IoT devices. Each device will rank a potential pair based on perceived latency and utilization, and are subsequently matched with their highest feasible choice. Although the matching algorithm itself is quick, node discovery followed by utilization and latency calculations for ranking cannot be done in real-time. Therefore, this method is ideal for small batch allocation.

When an IoT request is assigned to a fog node with insufficient available resources and is immediately propagated to Cloud, it is said that the request is *rejected* by the fog node. Assuming that fog may not have enough resources to service all IoT devices in the region, Intharawijitr et al. [31] formulate a model that minimizes the *blocking probability* of resource allocation, which is the average number of IoT workloads rejected by fog. For an IoT workload, three fog selection policies are explored: a *random* fog node, the fog node with *lowest-latency* to the application, or the fog node with *maximum available capacity*. Simulation results conclude the lowest-latency policy minimizes the blocking probability of IoT applications.

Skarlat et al. [9] formulate resource provisioning over the fog-Cloud system as an MILP to maximize fog utilization. The simulated evaluation concludes a decrease in response latency by 39% over default provisioning policies in CloudSim [95]. The same problem is formulated as a genetic algorithm with an evaluated increase of 150% cost of the MILP solution, and around 64% of applications assigned to fog.

Taneja and Davy [79] model each IoT application request as a set of dependent IoT tasks that may be processed by different fog nodes. The resource requirements of all current batch IoT tasks are sorted in ascending order, and a fog node satisfying the task resource requirements is selected for each task. By efficiently placing IoT applications on fog, increased fog utilization implicitly decreases the total response time, network usage and energy consumption compared to Cloud placement strategies. Karamoozian et al. [80] solve the resource allocation problem using a gravitational search algorithm. This meta-heuristic algorithm denotes each possible solution with a mass, and iteratively updates the particle masses based on their interactions with each other. At termination, the heaviest mass (solution) is selected.

Whereas [79, 80] represents each IoT application as a directed acyclic graph of IoT application tasks, Ni et al. [96] represent IoT tasks as a priced-timed Petri net which includes task transition time and price costs, in addition to task dependencies. The presented heuristic algorithm [96] allocates IoT applications to fog by prioritizing the minimization of response latency, followed by the maximization of fog *credibility* based on time cost, price cost, and resource capacity. The fog credibility is based on historical data of response rate, execution efficiency, reboot rate and reliability.

The focus of resource provisioning by Salaht et al. [97] is to first and foremost satisfy QoS requirements of IoT applications without further consideration of fog utility, resource cost or latency. Formulated as a Constraint Programming problem [97], it aims to create a scalable, generic and easily upgradeable placement service. Using the Smart Bell application [85] for comparison, the MILP model [85] executes in 343 seconds, whereas the Constrained Programming model [97] executes on the same application with the same parameters in 0.559 seconds without loss of solution QoS. We observe that considerations of cost optimality are ignored for a faster solution which may still prove too slow for real-time allocation of latency-sensitive IoT applications.

Donassolo et al. [98] introduce *Optimized Fog Service Provisioning* (O-FSP) resource allocation scheme to minimize IoT service cost and increase fog utilization. Based on available fog node resources and IoT QoS requirements, O-FSP follows a greedy approach to allocate IoT applications by resource cost. Although O-FSP succeeds in providing high fog infrastructure utilization and low resource costs, the execution time of O-FSP is not evaluated.

Gu et al. [28] support medical systems by reserving fog resources in the form of a VM assigned to the medical device. The resource provisioning problem is formulated as an MINLP, linearized, and reworked into a two-phase heuristic to minimize resource costs. Video streaming services can benefit from low latency of fog, but creates a carbon footprint as a result [4]. In response, Do et al. [4] formulate the resource allocation of these services as a convex optimization to minimize carbon footprint. They then develop an iterative heuristic inspired by proximal algorithms and Alternating Direction Method of Multipliers to serve video streaming services with guaranteed bandwidth and low carbon footprint. The *Mist* [17] crowd-sensing infrastructure also reserves VMs in fog for IoT use; the placement of VMs in fog are formulated as a linearized MINLP, and solved to allocate IoT with minimal VM deployment cost.

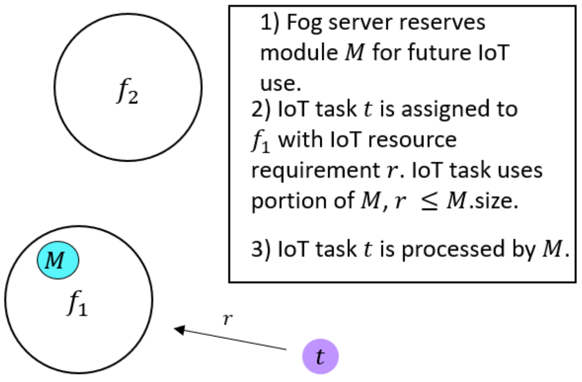
VI. Fog Computing Frameworks

Fog orchestration is defined by using a *control* layer to periodically monitor the available resources and request allocations to each fog node [99, 100]. Instead of an IoT request being uploaded directly to the fog layer, it is uploaded to the control layer which then distributes the application to fog nodes or Cloud accordingly. *Fog frameworks* can be seen as specific applications of fog orchestration to facilitate resource provisioning & allocation via a *fog orchestration controller* (FOC) or an API; the basic functionality of fog frameworks is shown in Fig. 6. If a module migration is required between fog nodes under the same FOC, the sending fog node will either consult the FOC to determine where to send the module, or send the module to the FOC for re-distribution. The fog layer may be partitioned into groups of fog nodes, each group with a separate FOC. In this case, a module migrated outside a cluster is sent to the FOC, which communicates with other FOCs to determine the module destination. If an API model is used for migration, a fog node will query other fog nodes to determine where to migrate data.

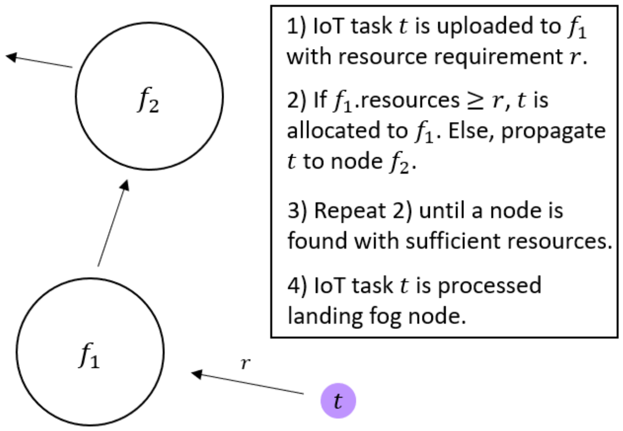
The frameworks covered in this section provide resource management of fog nodes via hardware and/or software extensions. By monitoring fog resource availability, a framework can distribute IoT requests to the fog without fog rejection, thus minimizing latency. In addition, certain frameworks allow data migration between fog nodes when an initial fog node has insufficient resources.

A. Non-fixed fog topologies

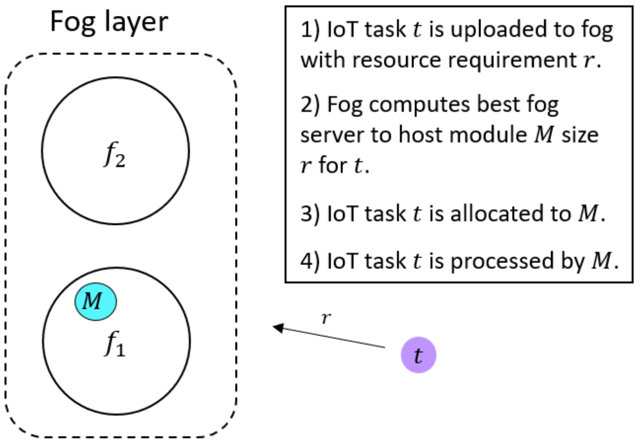
Chen et al. [38] define *fog-as-a-service-technology* (FA^{2}ST) as a fog framework that can support any IoT application, i.e. regardless of use case, with the objective of providing value-added fog services compared to Cloud. Importantly, FA^{2}ST provides on-demand fog service discovery [101], allowing it to probe all connected fog nodes for current resource availability at the moment an IoT request arrives. Furthermore, if an IoT application is assigned to a fog node that is no longer available, FA^{2}ST re-deploys the application to a new node. This allows FA^{2}ST to service a fog infrastructure with faulty, mobile, or dynamically available fog nodes. Madan et al. [102] envisions an IoV-fog infrastructure that provides UAVs to support overloaded RSUs. From a base station where all UAVs are kept and charged, an overloaded RSU triggers a UAV deployment to travel and hover over the RSU; data is migrated


1) Fog server reserves module M for future IoT use.
2) IoT task t is assigned to f_1 with IoT resource requirement r. IoT task uses portion of M, $r \leq M$.size.
3) IoT task t is processed by M.
f_2
M
f_1
r
t

(a) Prior provisioning and prompt allocation.

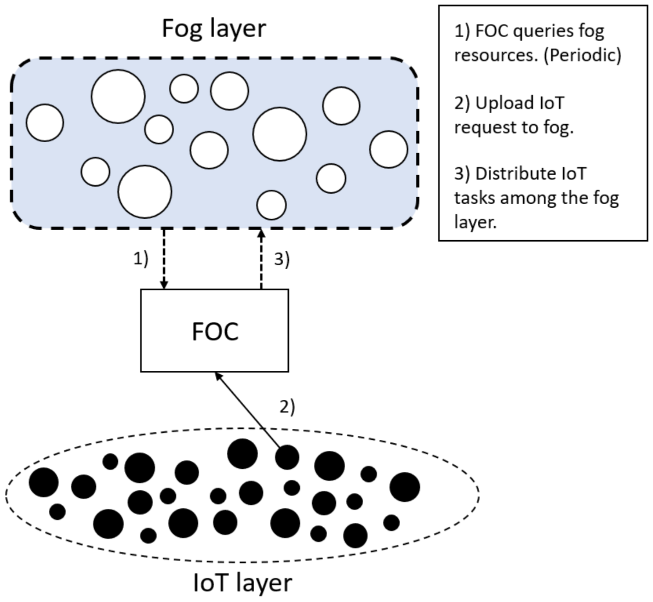

1) IoT task t is uploaded to f_1 with resource requirement r.
2) If f_1.resources $\geq r$, t is allocated to f_1. Else, propagate t to node f_2.
3) Repeat 2) until a node is found with sufficient resources.
4) IoT task t is processed landing fog node.
f_2
f_1
r
t

(b) On-demand provisioning and prompt allocation.

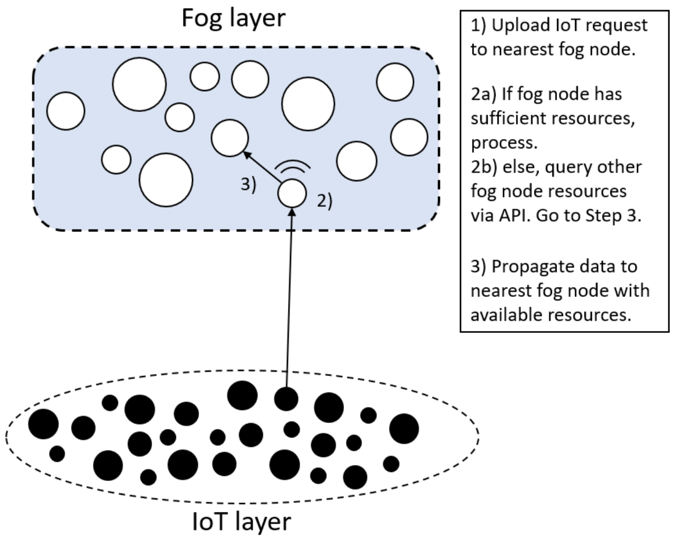

Fog layer
1) IoT task t is uploaded to fog with resource requirement r.
2) Fog computes best fog server to host module M size r for t.
3) IoT task t is allocated to M.
4) IoT task t is processed by M.
f_2
M
f_1
r
t

(c) On-demand provisioning and small batch allocation.

Figure 5: Resource management methods using a) prior provisioning and prompt allocation schemes, where fog resources are reserved for predicted IoT use, b) on-demand provisioning and prompt allocation schemes, where IoT task searches for suitable node, and c) on-demand provisioning and small batch allocation schemes, where optimal placement is found prior to processing. In b) and c), secondary node f_2 may be either a fog node or Cloud.


Fog layer
1) FOC queries fog resources. (Periodic)
2) Upload IoT request to fog.
3) Distribute IoT tasks among the fog layer.
1)
3)
FOC
2)
IoT layer

(a) FOC method.


Fog layer
1) Upload IoT request to nearest fog node.
2a) If fog node has sufficient resources, process.
2b) else, query other fog node resources via API. Go to Step 3.
3) Propagate data to nearest fog node with available resources.
3)
2)
IoT layer

(b) API method.

Figure 6: A fog framework provides current resource availability information of all fog nodes to streamline the resource allocation process. Two standard approaches are to use an FOC (hardware overhead) or to use an API (software overhead) to query fog and distribute IoT tasks.

to the UAV to accelerate the IoV computation and decrease response latency.

A framework can have a single FOC with a ubiquitous view of all fog resources [16, 98], or multiple FOCs, each monitoring a cluster of fog nodes [9, 29, 35]. A single FOC requires only one additional node; however it must have a connection to every fog node which may not be scalable for larger fog layers. Multiple FOCs provide improved scalability at a cost of increased hardware overhead. Furthermore, a module migration between FOCs may result in increased latency.

B. Fixed topologies

For a fixed fog topology, frameworks optimize the communication between end devices, fog, and Cloud. This approach to having an ubiquitous view of all fog information is through the use of an FOC which is connected to every fog either directly [9, 16, 43] or through APIs [17, 37, 39, 43, 56, 103, 104]. In addition to resource management for efficient IoT-fog service provisioning, most fog frameworks claim to be scalable to increased IoT traffic volume or fog infrastructure size [29, 35, 39, 43, 98].

Cardellini et al. [93] provide a distributed extension to the open source data stream processing application *Storm* to allow the execution of a distributed QoS-aware IoT resource scheduler. This extension allows fog nodes within the infrastructure to have knowledge of other fog node resource availability, and for IoT service scheduling to respect latency and resource requirements.

Donassolo et al. [98] propose *Fog-IoT ORchestator* (FITOR), a fog framework that monitors the fog infrastructure to survey and extract resource metrics from all fog nodes. This allows FITOR to have a ubiquitous view of all fog resources at any time, facilitating the service deployment of IoT data to fog nodes. As a result, FITOR is used in conjunction with the O-FSP resource provisioning scheme [98]. In order to allow fog nodes to accept any IoT request, FITOR must receive service descriptors from IoT applications along with the request. These descriptors define the IoT application, its building components and QoS requirements. Yigitoglu et al. [16] introduce the *Foggy* framework which accepts similar information from IoT, and deploys each task of an IoT request to a fog node with available resources and satisfying latency, privacy and priority requirements. The Foggy FOC monitors all fog resources via the MQTT protocol [105], and stores historical IoT requirements and workloads for faster future deployment planning. Although these frameworks aim to satisfy QoS of IoT requests by querying the requests, expecting requirement information from IoT may not be feasible.

Tuli et al. [35] propose *FogBus*, a scalable fog framework that partitions fog nodes into separate roles to increase security and reliability. FogBus is composed of fog gateway nodes, fog broker nodes (FOC), general computing nodes, and fog repository nodes. IoT applications upload data directly to fog gateway nodes, which are then forwarded to interconnected fog broker nodes, and distributed to fog general computing nodes for IoT request processing. Each fog broker node is also connected to a fog repository node which facilitates data sharing, replication, recovery and secured storage. Replication of stored data across multiple fog nodes provides storage robustness and reliability for possible fog node failures. Indeed, data stored on faulty fog brokers can be recovered by fog repository nodes, and replaced by existing fog broker nodes. FogBus also supports blockchain integration to verify that IoT data is coming from a pre-defined credible source.

Skarlat et al. [9] propose a fog architecture with groups of fog nodes clustered into *colonies* with all fog resource information available to an FOC. Upon receiving an IoT request, the FOC allocates IoT tasks among the fog nodes in the colony. If a fog colony does not have enough resources to process the IoT request, the FOC can query other colonies for available resources and transfer the request accordingly via REST APIs. It is recognized that not all IoT requests are suited for fog [9], e.g., non-delay sensitive requests or resource intensive big data. As a result, [9] provides Cloud-fog middleware to propagate these applications for Cloud resource allocation.

Zhang et al. [29] consider data migration between RSUs and fog nodes of their proposed IoV-fog application. Several fog nodes are clustered to provide seamless data sharing between nodes in a select region. Between clusters exists a *coordinator* (FOC) that manages fog resources over the system. If a fog cluster requires extra resources, or observes a vehicle moving towards a new region, it may handover an IoT module to a neighbouring fog cluster to continue uninterrupted IoT processing.

C. SDN-based frameworks

An SDN decouples the control plane and data planes of a traditional network, allowing an SDN controller to forward data through the network based on an arbitrary rule set [106]. Combining SDN with a standardized switch like OpenFlow [64] provides a simpler interface for SDN controllers to interact with the network. Furthermore, SDN can track mobile IoT devices to predict future destinations and support seamless data handover between fog nodes for uninterrupted IoT support [39].

Tomovic et al. [39] propose to achieve resource management, traffic control, and data migration of the fog plane via an SDN with OpenFlow controllers [64] which supports API functionality for IoT deployment. The Mist architectural framework [17] similarly uses SDN to monitor fog infrastructure resources, and APIs to monitor both IoT and fog device health, and facilitate IoT service allocation to fog.

Yousefpour et al. [43] propose *FogPlan*, a lightweight QoS-aware framework that uses an SDN controller to monitor incoming IoT traffic and fog node resources, and deploy IoT processing accordingly via an API. Notably, FogPlan aims to satisfy IoT QoS requirements with no or minimal IoT requirement information, increasing the workload of FogPlan while decreasing that of IoT devices. They use FogPlan to enable the min-cost and min-viol real-time resource provisioning scheme [43].

D. Data Migration

The *Foglet* [37] and *Mobile Fog* [103] programming models focus on module migration between IoT devices and fog nodes in a hierarchical architecture. They both implement an API that allows modules to either migrate one hierarchical level above or below, or to move to a specific fog node.

The data migration implementation with foglets [37] supports IoT requests that can simultaneously use different parts of the fog-Cloud infrastructure for different IoT tasks. Mobile Fog [103] creates a dynamic scaling mechanism whereby new on-demand fog nodes are created in response to overloaded workloads, and data is appropriately distributed to new fog nodes. Dynamic scaling with API data migration support creates a programming model well suited for mobile IoT devices such as smart cars, smart phones, and smart cameras.

Zhou et al. [47] propose a *vehicular fog computing* (VFC) framework with associated base stations to provide intra-fog resource management. Since a data user uploads their data to the nearest fog node, it is possible for a node to become overloaded during peak usage periods. In these cases, VFC allows for task offloading to nearby underutilized smart vehicles.

VII. Fog Infrastructure Design Evaluation

The efficiency or desirability of a fog computing infrastructure can be evaluated using simulation and emulation. Furthermore, simulating/emulating an existing fog infrastructure can be useful to fog application developers in deciding whether application processing should be done locally, by fog, or by Cloud [107]. There exists a variety of tools that deploy custom resource management policies, each focusing on different evaluation metrics and use case support. In the implementation of fog evaluation tools, the allocation and migration of modules are considered in terms of VMs. For more complex dynamic fog infrastructures, emulation frameworks can evaluate service of IoT applications in a way that more closely mimics the real fog infrastructure.

A. Simulation

In all simulation cases, a fog infrastructure is first defined manually by the user. Based on simulated IoT traffic parameters, the effects of network flow through fog can be evaluated. In most cases, the routing and interactions with fog nodes are defined by fog processing and forwarding policies. In the case of FogExplorer [108, 109], each IoT request is mapped to a selected machine for processing, and the effects of the deployment mapping on cost and QoS are evaluated. Each IoT request can be mapped to self, a specific fog node, or to Cloud. While designing an IoT application, testing different mapping options enables developers to identify the best location for IoT application processing [107].

iFogSim [36], an extension of CloudSim [95], is a simulation toolkit for IoT and fog that allows users to measure the resource cost, network use, energy consumption and latency of a specific network and resource management policy. Many resource management schemes use iFogSim to measure the impact of their proposed contributions [41, 34, 56, 110, 111]. EdgeNetworkCloudSim [112, 113] is also an extension of CloudSim that allows the simulation and evaluation of user-defined algorithms for placement, orchestration and consolidation of service chains. The system frequently monitors fog resources and energy consumption, and logs IoT task rejections due to insufficient fog resources. It does not however take into consideration the latency of IoT allocations.

Multiple tools extend iFogSim to address scenarios that cannot be simulated with iFogSim alone. iFogSimWithDataPlacement [110, 114] allows the implementation of resource management strategies that optimize data placement for a selected metric; this is achieved through MILP, and divide & conquer algorithm support. FogWorkflowSim [115] combines elements from WorkflowSim [116] and iFogSim to model and simulate workflows in fog. Identifying tasks that can be processed in parallel within a workflow decreases the overall latency and energy consumption [115]. Both MyiFogSim [117, 118] and MobFogSim [119] allow users to define the migration of VMs allocated to a mobile IoT device. The migration policy determines when a VM should be migrated by defining a migration zone around a Fog node, and a migration point an IoT device must cross to signal that it is moving away from the Fog node. The migration strategy defines where and how to migrate the VM. To where a VM should be migrated is dictated by the speed and direction of mobility, while the migration itself uses one of many strategies with different possible VM down time and memory usage during transfer. This scenario is shown in Fig. 7a.

FogNetSim++ [120, 121] extends fog support to the OMNet++ [122] discrete event simulator. The system uses a fog broker node to monitor fog resources, and to allocate IoT requests using a greedy approach by distance. Uniquely, FogNetSim++ allows for mobile fog node modelling. If a fog node moves away from an allocated IoT device, the resource module is migrated to the nearest fog node within the IoT device's range. This scenario is shown in Fig. 7b.

YAFS [123] is a discrete-event simulator that provides highly customizable placement, scheduling and routing strategies for IoT requests in fog. YAFS considers fog server failures, modeled using an exponential distribution. PureEdgeSim [124, 125] defines a total energy attribute for each fog server, along with energy consumption per processed IoT task. A fog server fails when its total energy is depleted. FogDirSim [126, 127] is a tool to simulate CISCO FogDirector, an IoT/fog manager. The simulator probabilistically predicts fog utilization, energy consumption, and robustness to fog node failures of a user-defined resource management policy. For these three simulators [123, 124, 126], a fog server failure triggers the re-allocation of modules present in the failed node are re-allocated. This scenario is shown in Fig. 7c.

PFogSim [91] defines a multi-tier fog architecture with Cloud in the final layer. The custom resource management policies define which layers can participate in IoT processing, and how IoT processes are distributed among the valid layers to optimize latency or operational cost. By restricting the scope of IoT assignment, a user can test various infrastructures to determine a minimal viable infrastructure. Therefore, PFogSim may serve as an empirical method for fog design using a multi-tier architecture.

Most evaluation tools use discrete-event simulation to simulate and evaluate the life-cycle of a fog infrastructure. In contrast, FogTorchΠ [128] uses Monte Carlo simulations to generate several possible IoT assignments that satisfy IoT QoS, hardware and software requirements. This approach allows a user to test resource allocation feasibility under several circumstances and requirements. The latency and bandwidth requirements from each simulation is analyzed to determine the set of resource deployments that yield the best estimated QoS.

B. Emulation

Simulation is a cost effective and computation efficient method of evaluating IoT interactions with a fog infrastructure; however, it usually assumes a number of simplifications that may not mimic a dynamic fog infrastructure well. In complex systems, *emulation* duplicates the fog topology and IoT workload, providing repeatable and controllable experiments of real IoT applications [129].

Héctor [130], MockFog [131, 132], and EmuFog [129, 133] are three fog emulation toolkits that aim to provide more realistic testing of fog infrastructures and IoT applications. In addition to emulating the fog infrastructure, Héctor also represents each IoT device as a VM with configurable properties, creating one emulated IoT environment. A *testbed* is a set of configured IoT request patterns, IoT resource requirements, and fog network conditions such as packet loss and random additional delay. Automated execution of various testbeds can therefore mimic the most realistic flow of IoT-fog interactions. MockFog allows users to inject failures into the fog system by toggling select fog nodes as unavailable. This will reduce fog resource availability in order to further test fog infrastructure resiliency and fault-tolerance. These three emulators allow users to fully define and design a fog infrastructure from scratch, though EmuFog also allows users to place fog nodes along the topology using a default greedy approach, or a custom placement policy. By changing the fog design, placement policies, or network conditions, users can evaluate the effects of the different fog infrastructures on IoT service. In this manner, emulation toolkits may serve as an empirical method for fog design.

VIII. Open Issues & Research Opportunities

A. Fog Design & Dimensioning

Both [5] and [78] are based on data traffic from static IoT devices; they assume a reliable fog infrastructure. In practice, many IoT devices are dynamic in location and request frequency, leading to possible spikes in incoming IoT traffic. Likewise, fog node failures will lead to spikes in incoming IoT traffic to active fog nodes. Updated simulations of fluctuating network congestion on the edge and stochastic fog node failures can serve useful to understanding the latency impact from fog. A fog design & dimensioning solution can be sensitive to network traffic, fog reliability, and fog modeling parameters such as candidate resource location and quantity. Therefore, a fog design & dimensioning solution should provide satisfactory QoS conditions to IoT under volatile IoT traffic and fog node failures, up to a degree of confidence.

In the event of a node failure, all IoT requests on the failed fog node should be re-allocated. One approach to ensure successful re-allocation is to implement the addition of repository nodes in the design of a fog infrastructure to backup fog node data. Adding mechanisms for IoT re-allocation will increase the fault-tolerance of the designed fog infrastructure.

The extensibility of [78] assumes more fog nodes are added for a static increase in IoT requests; however, if the added fog nodes increase the geographic range of the fog infrastructure, then previously out-of-range IoT devices may now be in range. The resulting total IoT traffic to the extended fog infrastructure may increase to more than estimated. IoT traffic patterns may evolve over time, and IoT traffic may reduce in certain areas and increase in others. An extension of current fog infrastructure should identify fog resources that can be repurposed to other fog nodes to optimize extension costs. Simulation of increased IoT traffic from geographic extensions to the fog infrastructure can be useful to understand fog extensibility beyond what is currently proposed.

These open issues provide an opportunity for future research, including the use of dynamically located and available fog servers, and fault-tolerance.

B. Fog Resource Provisioning & IoT Resource Allocation

In many cases, it is assumed that fog resources are previously known to IoT devices [4, 29, 43, 79, 85, 97] and often provided by an FOC with connections to every fog node [9, 98]. Some schemes assume a fog server in range of IoT satisfies latency [82, 98], or that current fog infrastructure follows certain architectural characteristics [14, 29, 89].

The standard approach with prompt allocation schemes is to upload IoT data to the nearest fog server, regardless of available fog resources or fog resource pricing. Indeed, it is assumed that this information is not known prior to IoT-fog connection. This can result in sub-optimal processing costs in a multi-price fog environment. If the nearest fog server is heavily congested with requests, the response latency can increase depending on whether the IoT request waits in a queue for processing [56], is propagated to different fog nodes [29], or is propagated to Cloud [30, 56]. This is shown in Fig. 5b. Although prior provisioning approaches [14, 81] help mitigate IoT requests being re-distributed elsewhere, resource cost concerns persist. As shown in Fig. 5a, a set of resources are reserved per fog node based on IoT traffic predictions. The fog node immediately allocates an incoming IoT request to the reserved resources, reducing latency; however, this assumes that reserved resources are always sufficient for IoT use. Another approach to mitigate re-distribution is to have a system-wide knowledge of fog resources [29, 43], however this generates additional hardware and/or software overhead.

The machine learning approach with prompt allocation schemes is to use RL to improve the efficiency of fog resource management over time [87, 88, 94]. However, QoS requirements of IoT requests are assumed to be met through the use of fog, and not verified. Currently, all current RL applications for fog resource management use a centralized RL agent. For large systems, the use of distributed multi-agent RL, each overseeing a fraction of total fog nodes, may improve system resource management as a whole. To reduce computation overhead, each RL agent should learn to independently behave optimally without requiring cooperation between them. Most resource management schemes assign a single IoT request to multiple modules in separate fog nodes without consideration of fog routing costs. Multiple multi-agent RL applications have been successful in decreasing network latency [134, 135], energy consumption [136] or cost [137] of general network routing between a source and target node. There has yet to be a distributed RL application for fog. Therefore, research opportunities exist for fog resource management using distributed RL for module assignment and inter-module routing.

Schemes that make an effort to provide QoS satisfying allocations generate small amounts of latency overhead [85] which may prove too large for certain IoT applications, or are currently only effective for IoV applications [84]. Future research is required into prompt allocation that does not jeopardize latency in the worst-case, and provides near-optimal resource allocation costs with little overhead. The objectives and modeling techniques of reviewed prompt resource allocation schemes are summarized in Table IV.

Small batch allocation schemes have longer resource provisioning & allocation solution times for better cost or latency, making them ideal for static IoT devices needing long-term and frequent fog support; however, mobile single-request IoT devices may also request resources from fog. Further research is required to provide resource provisioning & allocation for both static and dynamic IoT devices.

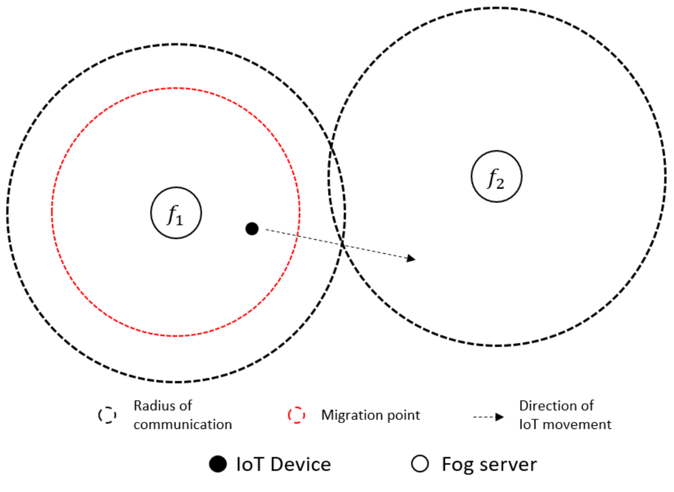
All small batch allocation schemes are determined to assume: 1) a static set of IoT requests for the duration of allocation optimization, 2) guaranteed bandwidth availability, and 3) all current fog resources are known. Each model is formulated to service IoT requests in batches, which may require additional latency for a sufficiently large batch to accumulate. The objectives and modeling techniques of reviewed small batch allocation schemes are summarized in Table V.

C. Fog Computing Frameworks

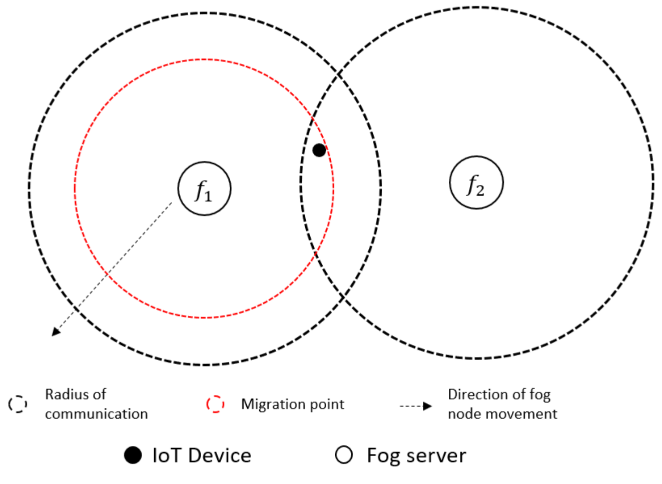
An ideal fog framework can monitor resources from the fog layer, and provide possible data migration between fog servers with minimal additional latency and overhead. Additional hardware must be able to communicate efficiently with existing fog infrastructure, which may be problematic if the framework and infrastructure are owned by different entities. Similarly, additional software such as API support must be installed on the individual fog nodes, implying the framework owner has access to the fog node software. An ideal FOC requires no additional information from IoT regarding the latency, privacy or security QoS, and can support any heterogeneous IoT request.

A single fog node provides privacy when data is stored locally; however, using an FOC that distributes an IoT request among a cluster of fog nodes can broaden the scope of data exposure to multiple fog nodes. This creates a trade-off between larger clusters of fog nodes under one FOC which provides allocation efficiency, and increased data privacy when using smaller clusters.

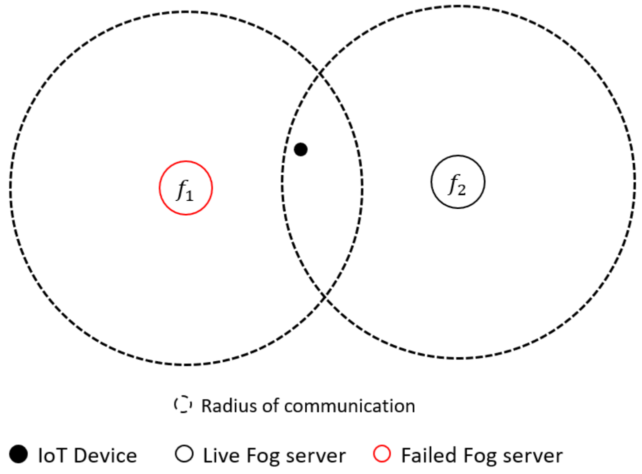
To the best of our knowledge, all mentioned fog frameworks assume ownership of the fog infrastructure, simplifying all interactions between the fog framework and the fog infrastructure. In a scenario with multiple public fog servers with different owners, a framework for each owner could create redundant overhead. Further research is needed to propose a fog framework that systematically addresses issues of generated overhead, fog infrastructure/framework ownership,



(a) Mobile IoT device



(b) Mobile Fog server



(c) Failed Fog server

Figure 7: Module migration due to: a) mobile IoT device moving between fog server ranges, b) mobile fog server moving out of IoT device range, and c) fog server failure. In all scenarios, the IoT task is initially connected to f_1, and is migrated to f_2. In a) and b), the IoT task begins migration once the IoT device crosses the migration point.

Table IV: Summary of resource provisioning schemes with prompt allocation.

Year	Author	Optimization objective	Modeling techniques
2015	Aazam and Huh [14, 81]	Fair resource pricing and prior provisioning based on historical IoT relinquish behaviour.	Formula to determine quantity of resources to allocate to IoT.
2016	Aazam et al. [82]	Extends prior provisioning of [14, 81] to factor QoE.	Formula to determine quantity of resource to allocate to IoT.
2016	Agarwal et al. [30]	Maximize fog utilization.	Efficient resource allocation algorithm.
2017	Zhang et al. [29]	Minimize energy-consumption of intra-fog resource management, minimize IoT service rejection of inter-fog resource management.	Convex optimization, and min-max optimization.
2017	Bittencourt et al. [56]	Minimize latency.	Evaluation of three IoT processing policies – concurrent, FIFO, delay-priority.
2018	Yousefpour et al. [43]	Minimize cost or delay violations.	Greedy algorithms with periodic execution.
2018	Xia et al. [85]	Minimize latency.	Anchor-based fog node ordering and dynamic component ordering heuristics.
2018	Wei et al. [87]	Minimize latency.	Reinforcement learning.
2018	Sun et al. [94]	Minimize power consumption of fog.	Reinforcement learning.
2018	Wang et al. [88]	Minimize latency.	Reinforcement learning.
2019	Aazam et al. [83]	Predicts and prior provisions resource consumption based on QoS and QoE.	Formula to determine quantity of resource to allocate to IoT.
2019	Zhou et al. [47]	Maximize vehicular fog utility.	Base station conducted pricing-based stable matching algorithm.
2020	Peng et al. [84]	Match vehicular fog nodes with IoT device based on resource, latency, reputation and pricing requirements of both parties.	One-to-one assignment algorithm to match vehicular fog nodes with clients.

Table V: Summary of resource provisioning schemes with small batch allocation.

Year	Author	Optimization objective	Modeling techniques
2015	Gu et al. [28]	Minimize cost.	MILP-based two-phase heuristic.
2015	Do et al. [4]	Minimize carbon footprint.	Distributed algorithm based on proximal algorithm and alternating direction method of multipliers.
2016	Zeng et al. [89]	Minimze latency.	MILP-based three-stage heuristics.
2016	Souza et al. [32]	Minimize cost.	MILP.
2016	Zhang et al. [90]	Minimize cost.	Stackelberg game between Fog nodes, MDCOs and IoT.
2016	Intharawijitr et al. [31]	Minimize blocking probability.	Evaluation of three IoT assignment policies — random, lowest latency fog node, maximum resource capacity Fog.
2016, 2017	Skarlat et al. [9, 138]	Maximize fog utilization.	MILP and genetic algorithm.
2017	Arkian et al. [17]	Minimize cost.	Linearized MINLP.
2017	Taneja and Davy [79]	Optimize latency, fog utilization and energy consumption.	ModuleMapping algorithm.
2017	Ni et al. [96]	1) Minimize latency, 2) maximize fog utility.	Heuristic algorithm.
2018	Ali et al. [92]	Minimize latency.	Many-to-one matching algorithm.
2019	Donassolo et al [98]	Minimize cost.	MILP model for exact solution, heuristic algorithm for real-time allocation.
2019	Salaht et al. [97]	Satisfy QoS.	Constraint Programming.
2019	Karamoozian et al. [80]	Minimize latency.	Gravitational Search Algorithm.

Table VI: Summary of contributions with fog frameworks and data migration.

			Overhead	
Year	**Author**	**Main Contribution**	**FOC**	**API**
2013	Hong et al. [103]	*Mobile fog* programming model for data migration between IoT and fog nodes.	No	Yes
2015	Gu et al. [28]	Fog infrastructure for medical cyber-physical systems with possible data migration among fog nodes.	Multiple	No
2015	Cardellini et al. [93]	Frequent resource availability sharing between fog nodes.	Single	No
2016	Saurez et al. [37]	*Foglet* programming model for data migration between IoT and fog nodes.	No	Yes
2016	Skarlat et al. [9, 138]	Groups fog nodes into colonies, each with a data migration controller to transfer IoT data between colonies if current fog colony resources are insufficient.	Multiple	Yes
2017	Yigitoglu et al. [16]	*Foggy* fog framework allows ubiquitous view of all fog resources, deploys IoT to fog nodes satisfying latency, privacy and priority requirements.	Single	No
2017	Zhang et al. [29]	Data migration between RSUs and fog nodes.	Multiple	No
2017	Tomovic et al. [39]	Resource management, traffic control and data migration via SDN.	No	Yes
2018	Yousefpour et al. [43]	*FogPlan* lightweight QoS-aware framework using SDN to control and monitor incoming IoT traffic and fog resources.	No	Yes
2018	Chen et al. [38]	*FA^2ST*, infrastructure of cross-domain supporting fog nodes.	Single	No
2019	Zhou et al. [47]	Vehicular fog framework with intra-fog resource management. If a fog becomes overloaded, allows task offloading to nearby smart vehicles.	Multiple	No
2019	Tuli et al. [35]	*FogBus* fog framework partitions fog nodes into separate roles and IoT exposure to increase fog security and reliability.	Multiple	No
2019	Donassolo et al. [98]	*FITOR*, IoT-fog orchestration framework that allows ubiquitous view of all fog resources.	Single	No
2020	Madan et al. [102]	Allows overloaded RSUs to call and offload data to UAVs.	Multiple	No
2020	Peng et al. [84]	Multi-attribute double auction vehicular framework to match and pair vehicular fog nodes with IoT devices based on resources, latency, reputation and pricing requirements of both parties.	Multiple	No

Table VII: Summary of discrete-event simulation and emulation toolkits for Fog computing infrastructure — Functionality

			Migration of IoT process due to			
Year	**Name**	**Method**	**Mobile IoT**	**Fog node failure**	**Mobile Fog node**	**Topology input**
2017	EmuFog [129, 133]	Emulation	No	No	No	BRITE [139]
2017	iFogSim [36, 140]	Simulation	No	No	No	GUI, JSON
2017	MyiFogSim [117, 118]	Simulation	Yes	No	No	GUI, JSON
2017	EdgeNetworkCloudSim [112, 113]	Simulation	No	No	No	BRITE [139]
2018	FogExplorer [108, 109]	Simulation	No	No	No	GUI
2018	iFogSimWithDataPlacement [110, 114]	Simulation	No	No	No	GUI, JSON
2018	FogNetSim++ [120, 121]	Simulation	No	No	Yes	.INI file
2019	PFogSim [91]	Simulation	No	No	No	XML
2019	YAFS [123]	Simulation	No	Yes	No	JSON
2019	PureEdgeSim [124]	Simulation	No	Yes	No	Java
2019	FogWorkflowSim [115]	Simulation	No	No	No	Java
2019	Héctor [130]	Emulation	No	No	No	API
2019	MockFog [131, 132]	Emulation	No	No	No	API, GUI
2020	FogDirSim [126, 127]	Simulation	No	Yes	No	Database
2020	MobFogSim [119]	Simulation	Yes	No	No	GUI, JSON

Table VIII: Summary of discrete-event simulation and emulation toolkits for Fog computing infrastructure – Evaluation Criteria

			Evaluated Metrics				
Year	**Name**	**Method**	**Latency**	**Operational Costs**	**Energy consumption**	**Fog utilization/ congestion**	**Task Rejection**
2017	EmuFog [129, 133]	Emulation	Yes	No	Yes	Yes	No
2017	iFogSim [36, 140]	Simulation	Yes	Yes	Yes	Yes	No
2017	MyiFogSim [117, 118]	Simulation	Yes	Yes	Yes	Yes	No
2017	EdgeNetworkCloudSim [112, 113]	Simulation	No	No	Yes	Yes	Yes
2018	FogExplorer [108, 109]	Simulation	Yes	Yes	No	No	No
2018	iFogSimWithDataPlacement [110, 114]	Simulation	Yes	Yes	Yes	Yes	No
2018	FogNetSim++ [120, 121]	Simulation	Yes	No	Yes	No	Yes
2019	PFogSim [91]	Simulation	Yes	Yes	No	No	Yes
2019	YAFS [123]	Simulation	Yes	No	No	Yes	Yes
2019	PureEdgeSim [124]	Simulation	Yes	No	Yes	Yes	No
2019	FogWorkflowSim [115]	Simulation	Yes	Yes	Yes	No	No
2019	Héctor [130]	Emulation	Yes	No	No	Yes	No
2019	MockFog [131, 132]	Emulation	Yes	No	No	Yes	Yes
2020	FogDirSim [126, 127]	Simulation	No	No	Yes	No	Yes
2020	MobFogSim [119]	Simulation	Yes	Yes	Yes	Yes	No

privacy, and communication with IoT applications. There is also a research opportunity to develop a proof-of-concept of a framework that can interact with existing public fog infrastructure regardless of owner. The main contributions and hardware/software overhead generated for each fog framework is summarized in Table VI.

D. Fog Infrastructure Design Evaluation

After an IoT request is assigned to a fog node, there are several scenarios in which the provisioned module needs to be migrated to a new fog node. Specifically, we consider module migration due to a mobile IoT device moving between fog nodes, a mobile fog node moving away from an IoT device, and a fog node failure. When a mobile IoT moves between fog nodes, module migration is efficient for slow moving devices; however, significant decreases in successful module migrations is seen with fast vehicles [119]. The proposed fog implementations with mobile fog nodes use slow moving UAVs [141] and buses [46], ensuring migration efficiency. Though module handover is supported in FogNetSim++ [120], specifics regarding migration policies and strategies are not fully explored beyond what is available in OMNet++ [122]. When fog node failures are present, it is necessary for replicas of the module data to be stored elsewhere in order to restore and migrate data to live fog nodes [35].

For all current simulators, at most one of these scenarios are explicitly addressed. Though fog computing has many use cases that require module migration, there is yet to be a tool that simulates all possible scenarios. For each described evaluation tool, the approach to simulation/emulation is only appropriate if the designed fog infrastructure and framework can perform similar actions.

From an implementation standpoint, file input support is ideal in situations where the entities designing the fog infrastructure and evaluating its efficiency are different. Using formats like JSON, XML or BRITE [139] can facilitate the fog development process across multiple entities in an organization. A GUI or topology input directly into the evaluation tool can be useful when trial-and-error is required to find an optimal design. We wish to mitigate this using a fog design & dimensioning process [5, 78]. The supported migration scenarios and topological input format of each simulator is summarized in Table VII.

Each tool records logs during evaluation, outlining metrics of each task allocation such as latency, energy consumption, cost of allocation, resulting network congestion, and task rejection due to unsatisfied QoS requirements. Though most tools allow custom output metrics, only what is recorded in logs can be evaluated; indeed, no evaluation tool records all evaluation metrics shown in Table VIII.

There remain opportunities to build an evaluation tool with the flexibility to model any scenario in a fog infrastructure. This becomes particularly important in a dynamic fog system with fog node failures, dynamically available fog nodes, mobile fog nodes, and mobile IoT. Since the evaluated output can only aggregate what is logged, there is also an opportunity to expand the available metrics that are recorded from the IoT request, the provisioned fog node, and the network path.

IX. Conclusion

Fog computing provides an alternative to Cloud processing with increased privacy and low latency to IoT applications. As the number of IoT applications increases, fog infrastructure implementations become crucial. We have identified four phases to implement a fog infrastructure, and discussed their limitations and open issues. We compared current design & dimensioning models which produce a blueprint of a fog infrastructure for the support of IoT applications. We classified resource provisioning & allocation schemes by their effectiveness to support dynamic and static IoT applications, and identified optimal objectives and modelling techniques used by each scheme. We analyzed the main contributions and generated overhead of fog frameworks. Finally, we reviewed and identify limitations of simulation/emulation tools for the evaluation of the designed fog infrastructure.

With this survey of current fog solutions, we intend to give a detailed understanding of the necessary considerations of building a practical fog infrastructure for local IoT support, up to large-scale Smart City systems. Based on our results, we believe we have provided a detailed overview of steps necessary for a practical and functional fog implementation.

References

[1] IDC. *The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast*. Tech. rep. Accessed: 2020-04. 2019. URL: https://www.idc.com/getdoc.jsp?containerId=prUS45213219.

[2] CISCO. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. Tech. rep. Accessed: 2019-04. 2015. URL: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[3] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn, et al. "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures". In: *Journal of Cloud Computing* 7.1 (2018), p. 16.

[4] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong. "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing". In: *2015 International Conference on Information Networking (ICOIN)*. IEEE. 2015, pp. 324–329.

[5] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He. "Deployment and Dimensioning of Fog Computing-Based Internet of Vehicle Infrastructure for Autonomous Driving". In: *IEEE Internet of Things Journal* 6.1 (2019), pp. 149–160.

[6] S. W. Loke. "The internet of flying-things: Opportunities and challenges with airborne fog computing and mobile cloud in the clouds". In: *arXiv preprint arXiv:1507.04492* (2015).

[7] L. M. Vaquero and L. Rodero-Merino. "Finding your way in the fog: Towards a comprehensive definition of fog computing". In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 27–32.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog computing and its role in the internet of things". In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM. 2012, pp. 13–16.

[9] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner. "Resource provisioning for IoT services in the fog". In: *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*. IEEE. 2016, pp. 32–39.

[10] A. Giordano, G. Spezzano, and A. Vinci. "Smart agents and fog computing for smart city applications". In: *International Conference on Smart Cities*. Springer. 2016, pp. 137–146.

[11] R. Mahmud, R. Kotagiri, and R. Buyya. "Fog computing: A taxonomy, survey and future directions". In: *Internet of everything*. Springer, 2018, pp. 103–130.

[12] A. Ahmed, H. Arkian, D. Battulga, A. J. Fahs, M. Farhadi, D. Giouroukis, A. Gougeon, F. O. Gutierrez, G. Pierre, P. R. Souza Jr, et al. "Fog Computing Applications: Taxonomy and Requirements". In: *arXiv preprint arXiv:1907.11621* (2019).

[13] K. Bachmann. "Design and implementation of a fog computing framework". Diploma Thesis. Vienna University of Technology (TU Wien), Vienna, Austria, 2017.

[14] M. Aazam and E.-N. Huh. "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT". In: *2015 IEEE 29th International*

Conference on Advanced Information Networking and Applications. IEEE. 2015, pp. 687–694.

[15] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya. "Fog computing: Principles, architectures, and applications". In: *Internet of things*. Elsevier, 2016, pp. 61–75.

[16] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig. "Foggy: a framework for continuous automated IoT application deployment in fog computing". In: *2017 IEEE International Conference on AI & Mobile Services (AIMS)*. IEEE. 2017, pp. 38–45.

[17] H. R. Arkian, A. Diyanat, and A. Pourkhalili. "MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications". In: *Journal of Network and Computer Applications* 82 (2017), pp. 152–165.

[18] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara. "Cloudpath: A multi-tier cloud computing framework". In: *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 2017, pp. 1–13.

[19] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. "Edge computing: Vision and challenges". In: *IEEE internet of things journal* 3.5 (2016), pp. 637–646.

[20] M. Satyanarayanan. "The emergence of edge computing". In: *Computer* 50.1 (2017), pp. 30–39.

[21] K. Dolui and S. K. Datta. "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing". In: *2017 Global Internet of Things Summit (GIoTS)*. IEEE. 2017, pp. 1–6.

[22] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The case for vm-based cloudlets in mobile computing". In: *IEEE pervasive Computing* 8.4 (2009), pp. 14–23.

[23] E. Zygmuntowicz, V. Spivak, K. Skaar, D. Collison, O. Shaldybin, M. Lucovsky, and K. Murphy. *Microcloud platform delivery system*. US Patent 8,813,065. Aug. 2014.

[24] H. T. Dinh, C. Lee, D. Niyato, and P. Wang. "A survey of mobile cloud computing: architecture, applications, and approaches". In: *Wireless communications and mobile computing* 13.18 (2013), pp. 1587–1611.

[25] M. T. Beck, M. Werner, S. Feld, and S. Schimper. "Mobile edge computing: A taxonomy". In: *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer. 2014, pp. 48–55.

[26] A. Enayet, M. A. Razzaque, M. M. Hassan, A. Alamri, and G. Fortino. "A mobility-aware optimal resource allocation architecture for big data task execution on mobile cloud in smart cities". In: *IEEE Communications Magazine* 56.2 (2018), pp. 110–117.

[27] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani. "An open ecosystem for mobile-cloud convergence". In: *IEEE Communications Magazine* 53.3 (2015), pp. 63–70.

[28] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang. "Cost efficient resource management in fog computing supported medical cyber-physical system". In: *IEEE Transactions on Emerging Topics in Computing* 5.1 (2015), pp. 108–119.

[29] W. Zhang, Z. Zhang, and H.-C. Chao. "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management". In: *IEEE Communications Magazine* 55.12 (2017), pp. 60–67.

[30] S. Agarwal, S. Yadav, and A. K. Yadav. "An efficient architecture and algorithm for resource provisioning in fog computing". In: *International Journal of Information Engineering and Electronic Business* 8.1 (2016), p. 48.

[31] K. Intharawijitr, K. Iida, and H. Koga. "Analysis of fog model considering computing and communication latency in 5G cellular networks". In: *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. IEEE. 2016, pp. 1–4.

[32] V. B. C. Souza, W. Ramırez, X. Masip-Bruin, E. Marın-Tordera, G. Ren, and G. Tashakor. "Handling service allocation in combined fog-cloud scenarios". In: *2016 IEEE international conference on communications (ICC)*. IEEE. 2016, pp. 1–5.

[33] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang. "A hierarchical distributed fog computing architecture for big data analysis in smart cities". In: *Proceedings of the ASE BigData & SocialInformatics 2015*. ACM. 2015, p. 28.

[34] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand. "iFogStor: an IoT data placement strategy for fog infrastructure". In: *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. IEEE. 2017, pp. 97–104.

[35] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya. "FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing". In: *Journal of Systems and Software* (2019).

[36] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments". In: *Software: Practice and Experience* 47.9 (2017), pp. 1275–1296.

[37] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder. "Incremental deployment and migration of geo-distributed situation awareness applications in the fog". In: *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*. ACM. 2016, pp. 258–269.

[38] N. Chen, Y. Yang, T. Zhang, M.-T. Zhou, X. Luo, and J. K. Zao. "Fog as a service technology". In: *IEEE Communications Magazine* 56.11 (2018), pp. 95–101.

[39] S. Tomovic, K. Yoshigoe, I. Maljevic, and I. Radusinovic. "Software-defined fog network architecture for IoT". In: *Wireless Personal Communications* 92.1 (2017), pp. 181–196.

[40] X. Sun and N. Ansari. “EdgeIoT: Mobile edge computing for the Internet of Things”. In: *IEEE Communications Magazine* 54.12 (2016), pp. 22–29.

[41] S. S. Gill, R. C. Arya, G. S. Wander, and R. Buyya. “Fog-Based Smart Healthcare as a Big Data and Cloud Service for Heart Patients Using IoT”. In: *International Conference on Intelligent Data Communication Technologies and Internet of Things*. Springer. 2018, pp. 1376–1383.

[42] I. Stojmenovic. “Fog computing: A cloud to the ground support for smart things and machine-to-machine networks”. In: *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*. IEEE. 2014, pp. 117–122.

[43] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue. “Qos-aware dynamic fog service provisioning”. In: *arXiv preprint arXiv:1802.00800* (2018).

[44] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen. “Vehicular fog computing: A viewpoint of vehicles as the infrastructures”. In: *IEEE Transactions on Vehicular Technology* 65.6 (2016), pp. 3860–3873.

[45] M. Sookhak, F. R. Yu, Y. He, H. Talebian, N. S. Safa, N. Zhao, M. K. Khan, and N. Kumar. “Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing”. In: *IEEE Vehicular Technology Magazine* 12.3 (2017), pp. 55–64.

[46] G. Sun, S. Sun, H. Yu, and M. Guizani. “Towards Incentivizing Fog-Based Privacy-Preserving Mobile Crowdsensing in the Internet of Vehicles”. In: *IEEE Internet of Things Journal* (2019).

[47] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez. “Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach”. In: *IEEE Transactions on Vehicular Technology* 68.4 (2019), pp. 3113–3125.

[48] M. Ahmad, M. B. Amin, S. Hussain, B. H. Kang, T. Cheong, and S. Lee. “Health Fog: a novel framework for health and wellness applications”. In: *The Journal of Supercomputing* 72.10 (2016), pp. 3677–3695.

[49] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg. “Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach”. In: *Future Generation Computer Systems* 78 (2018), pp. 641–658.

[50] R. Sofia and P. Mendes. “User-provided networks: consumer as provider”. In: *IEEE Communications Magazine* 46.12 (2008), pp. 86–91.

[51] C. Chang, S. N. Srirama, and R. Buyya. “Indie fog: An efficient fog-computing infrastructure for the internet of things”. In: *Computer* 50.9 (2017), pp. 92–98.

[52] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy. “Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study”. In: *IEEE Access* 5 (2017), pp. 9882–9910.

[53] F. Y. Okay and S. Ozdemir. “A fog computing based smart grid model”. In: *2016 international symposium on networks, computers and communications (ISNCC)*. IEEE. 2016, pp. 1–6.

[54] C. Perera, D. S. Talagala, C. H. Liu, and J. C. Estrella. “Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in IoT clouds”. In: *IEEE Transactions on Computational Social Systems* 2.4 (2015), pp. 171–181.

[55] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud. “SmartCityWare: A service-oriented middleware for cloud and fog enabled smart city services”. In: *Ieee Access* 5 (2017), pp. 17576–17588.

[56] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar. “Mobility-aware application scheduling in fog computing”. In: *IEEE Cloud Computing* 4.2 (2017), pp. 26–35.

[57] S. Jeong, O. Simeone, and J. Kang. “Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning”. In: *IEEE Transactions on Vehicular Technology* 67.3 (2017), pp. 2049–2063.

[58] F. Zhou, Y. Wu, H. Sun, and Z. Chu. “UAV-enabled mobile edge computing: Offloading optimization and trajectory design”. In: *2018 IEEE International Conference on Communications (ICC)*. IEEE. 2018, pp. 1–6.

[59] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud. “UAVFog: A UAV-based fog computing for Internet of Things”. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2017, pp. 1–8.

[60] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos. “Fog computing for sustainable smart cities: A survey”. In: *ACM Computing Surveys (CSUR)* 50.3 (2017), p. 32.

[61] C. Dsouza, G.-J. Ahn, and M. Taguinod. “Policy-driven security management for fog computing: Preliminary framework and a case study”. In: *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*. IEEE. 2014, pp. 16–23.

[62] J. Liu, J. Li, L. Zhang, F. Dai, Y. Zhang, X. Meng, and J. Shen. “Secure intelligent traffic light control using fog computing”. In: *Future Generation Computer Systems* 78 (2018), pp. 817–824.

[63] S. Shin and G. Gu. “CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)” In: *2012 20th IEEE international conference on network protocols (ICNP)*. IEEE. 2012, pp. 1–6.

[64] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and

J. Turner. “OpenFlow: enabling innovation in campus networks”. In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74.

[65] S. Yi, Z. Qin, and Q. Li. “Security and privacy issues of fog computing: A survey”. In: *International conference on wireless algorithms, systems, and applications*. Springer. 2015, pp. 685–695.

[66] P. Hu, S. Dhelim, H. Ning, and T. Qiu. “Survey on fog computing: architecture, key technologies, applications and open issues”. In: *Journal of Network and Computer Applications* 98 (2017), pp. 27–42.

[67] M. Mukherjee, L. Shu, and D. Wang. “Survey of fog computing: Fundamental, network applications, and research challenges”. In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 1826–1857.

[68] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos. “A comprehensive survey on fog computing: State-of-the-art and research challenges”. In: *IEEE Communications Surveys & Tutorials* 20.1 (2017), pp. 416–464.

[69] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian. “Resource management approaches in fog computing: a comprehensive review”. In: *Journal of Grid Computing* (2019), pp. 1–42.

[70] A. Brogi, S. Forti, C. Guerrero, and I. Lera. “How to place your apps in the fog: State of the art and open challenges”. In: *Software: Practice and Experience* 50.5 (2020), pp. 719–740.

[71] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan. “Fog Computing: Survey of trends, architectures, requirements, and research directions”. In: *IEEE Access* 6 (2018), pp. 47980–48009.

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

[73] A. Markus and A. Kertesz. “A survey and taxonomy of simulation environments modelling fog computing”. In: *Simulation Modelling Practice and Theory* 101 (2020), p. 102042.

[74] M. Jia, J. Cao, and W. Liang. “Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks”. In: *IEEE Transactions on Cloud Computing* 5.4 (2015), pp. 725–737.

[75] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo. “Efficient algorithms for capacitated cloudlet placements”. In: *IEEE Transactions on Parallel and Distributed Systems* 27.10 (2015), pp. 2866–2880.

[76] A. Ceselli, M. Premoli, and S. Secci. “Cloudlet network design optimization”. In: *2015 IFIP Networking Conference (IFIP Networking)*. IEEE. 2015, pp. 1–9.

[77] Q. Fan and N. Ansari. “Cost aware cloudlet placement for big data processing at the edge”. In: *2017 IEEE International Conference on Communications (ICC)*. IEEE. 2017, pp. 1–6.

[78] I. Martinez, A. Jarray, and A. S. Hafid. “Scalable Design and Dimensioning of Fog-Computing Infrastructure to Support Latency Sensitive IoT Applications”. In: *IEEE Internet of Things Journal* 7.6 (2020), pp. 5504–5520.

[79] M. Taneja and A. Davy. “Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm”. In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 1222–1228.

[80] A. Karamoozian, A. Hafid, and E. M. Aboulhamid. “On the Fog-Cloud Cooperation: How Fog Computing can address latency concerns of IoT applications”. In: *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE. 2019, pp. 166–172.

[81] M. Aazam and E.-N. Huh. “Dynamic resource provisioning through Fog micro datacenter”. In: *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*. IEEE. 2015, pp. 105–110.

[82] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris. “MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT”. In: *2016 23rd International Conference on Telecommunications (ICT)*. IEEE. 2016, pp. 1–5.

[83] M. Aazam, K. A. Harras, and S. Zeadally. “Fog computing for 5G tactile industrial Internet of Things: QoE-aware resource allocation model”. In: *IEEE Transactions on Industrial Informatics* 15.5 (2019), pp. 3085–3092.

[84] X. Peng, K. Ota, and M. Dong. “Multi-attribute based Double Auction Towards Resource Allocation in Vehicular Fog Computing”. In: *IEEE Internet of Things Journal* (2020).

[85] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, and F. Desprez. “Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed iot applications in the fog”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. ACM. 2018, pp. 751–760.

[86] L. P. Kaelbling, M. L. Littman, and A. W. Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[87] Y. Wei, F. R. Yu, M. Song, and Z. Han. “Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning”. In: *IEEE Internet of Things Journal* 6.2 (2018), pp. 2061–2073.

[88] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo. “Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications”. In: *IEEE Transactions on Industrial Informatics* 15.2 (2018), pp. 976–986.

[89] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu. “Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system”. In: *IEEE Transactions on Computers* 65.12 (2016), pp. 3702–3712.

[90] H. Zhang, Y. Xiao, S. Bu, D. Niyato, R. Yu, and Z. Han. "Fog computing in multi-tier data center networks: A hierarchical game approach". In: *2016 IEEE international conference on communications (ICC)*. IEEE. 2016, pp. 1–6.

[91] Q. Wang. "PFogSim: A Simulator for Evaluating Dynamic and Layered Fog Computing Environments". In: (2019).

[92] M. Ali, N. Riaz, M. I. Ashraf, S. Qaisar, and M. Naeem. "Joint cloudlet selection and latency minimization in fog networks". In: *IEEE Transactions on Industrial Informatics* 14.9 (2018), pp. 4055–4063.

[93] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli. "On QoS-aware scheduling of data stream applications over fog computing infrastructures". In: *2015 IEEE Symposium on Computers and Communication (ISCC)*. IEEE. 2015, pp. 271–276.

[94] Y. Sun, M. Peng, and S. Mao. "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks". In: *IEEE Internet of Things Journal* 6.2 (2018), pp. 1960–1971.

[95] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". In: *Software: Practice and experience* 41.1 (2011), pp. 23–50.

[96] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu. "Resource allocation strategy in fog computing based on priced timed petri nets". In: *ieee internet of things journal* 4.5 (2017), pp. 1216–1228.

[97] F. A. Salaht, F. Desprez, A. Lebre, C. Prud'Homme, and M. Abderrahim. "Service Placement in Fog Computing Using Constraint Programming". In: *IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING*. 2019.

[98] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos. "Fog Based Framework for IoT Service Provisioning". In: *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2019, pp. 1–6.

[99] Y. Jiang, Z. Huang, and D. H. Tsang. "Challenges and solutions in fog computing orchestration". In: *IEEE Network* 32.3 (2018), pp. 122–129.

[100] M. S. de Brito, S. Hoque, T. Magedanz, R. Steinke, A. Willner, D. Nehls, O. Keils, and F. Schreiner. "A service orchestration architecture for fog-enabled infrastructures". In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE. 2017, pp. 127–132.

[101] N. Chen and S. Clarke. "A dynamic service composition model for adaptive systems in mobile computing environments". In: *International Conference on Service-Oriented Computing*. Springer. 2014, pp. 93–107.

[102] N. Madan, A. W. Malik, A. U. Rahman, and S. D. Ravana. "On-demand resource provisioning for vehicular networks using flying fog". In: *Vehicular Communications* (2020), p. 100252.

[103] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. "Mobile fog: A programming model for large-scale applications on the internet of things". In: *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. 2013, pp. 15–20.

[104] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos. "A platform as-a-service for hybrid cloud/fog environments". In: *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2016, pp. 1–7.

[105] D. Soni and A. Makwana. "A survey on mqtt: a protocol of internet of things (iot)". In: *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*. 2017.

[106] A. Lara, A. Kolasani, and B. Ramamurthy. "Simplifying network management using software defined networking and OpenFlow". In: *2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS)*. IEEE. 2012, pp. 24–29.

[107] J. Hasenburg, S. Werner, and D. Bermbach. "Supporting the evaluation of fog-based IoT applications during the design phase". In: *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things*. 2018, pp. 1–6.

[108] J. Hasenburg, S. Werner, and D. Bermbach. "FogExplorer". In: *Proceedings of the 19th International Middleware Conference (Posters)*. 2018, pp. 1–2.

[109] J. Hasenburg. *FogExplorer Inrastructure as Code*. https : / / github . com / OpenFogStack / FogExplorer. Accessed: 2020-07-25.

[110] M. I. Naas, J. Boukhobza, P. R. Parvedy, and L. Lemarchand. "An extension to ifogsim to enable the design of data placement strategies". In: *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. IEEE. 2018, pp. 1–8.

[111] A. Khanna and R. Tomar. "IoT based interactive shopping ecosystem". In: *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*. IEEE. 2016, pp. 40–45.

[112] M. Seufert, B. K. Kwam, F. Wamser, and P. Tran-Gia. "Edgenetworkcloudsim: Placement of service chains in edge clouds using NetworkCloudSim". In: *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2017, pp. 1–6.

[113] B. K. Kwam. *An extension of NetworkCloudSim to simulate the Edge Cloud*. https://github.com/lsinfo3/EdgeNetworkCloudSim. Accessed: 2020-05-02.

[114] M. I. Naas. *iFogSimWithDataPlacement on GitHub*. https : / / github . com / medislam / iFogSimWithDataPlacement. Accessed: 2020-05-02.

[115] X. Liu, L. Fan, J. Xu, X. Li, L. Gong, J. Grundy, and Y. Yang. "FogWorkflowSim: An Automated Simulation Toolkit for Workflow Performance Evaluation

in Fog Computing". In: *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2019, pp. 1114–1117.

[116] W. Chen and E. Deelman. "Workflowsim: A toolkit for simulating scientific workflows in distributed environments". In: *2012 IEEE 8th International Conference on E-Science*. IEEE. 2012, pp. 1–8.

[117] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt. "Myifogsim: A simulator for virtual machine migration in fog computing". In: *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*. 2017, pp. 47–52.

[118] M. M. Lopes. *MyiFogSim on GitHub*. https://github.com/marciocomp/myifogsim. Accessed: 2020-04-30.

[119] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt. "MobFogSim: Simulation of mobility and migration for fog computing". In: *Simulation Modelling Practice and Theory* 101 (2020), p. 102062.

[120] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan. "FogNetSim++: A toolkit for modeling and simulation of distributed fog environment". In: *IEEE Access* 6 (2018), pp. 63570–63583.

[121] T. Qayyum. *A Toolkit to simulate distributed fog computing environment*. https://github.com/rtqayyum/fognetsimpp. Accessed: 2020-04-30.

[122] A. Varga. "OMNeT++". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.

[123] I. Lera, C. Guerrero, and C. Juiz. "YAFS: A simulator for IoT scenarios in fog computing". In: *IEEE Access* 7 (2019), pp. 91745–91758.

[124] C. Mechalikh, H. Taktak, and F. Moussa. "PureEdgeSim: A Simulation Toolkit for Performance Evaluation of Cloud, Fog, and Pure Edge Computing Environments". In: *The 2019 International Conference on High Performance Computing Simulation*. IEEE, 2019, pp. 700–707.

[125] C. Mechalikh. *PureEdgeSim: A Simulation Toolkit for Performance Evaluation of Cloud, Fog, and Pure Edge Computing Environments*. https://github.com/CharafeddineMechalikh/PureEdgeSim. Accessed: 2020-04-30.

[126] S. Forti, A. Pagiaro, and A. Brogi. "Simulating FogDirector Application Management". In: *Simulation Modelling Practice and Theory* 101 (2020), p. 102021.

[127] S. Forti. *A Simulator of CISCO FogDirector Application Management*. https://github.com/di-unipi-socc/FogDirSim. Accessed: 2020-05-02.

[128] A. Brogi, S. Forti, and A. Ibrahim. "How to best deploy your Fog applications, probably". In: *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. IEEE. 2017, pp. 105–114.

[129] R. Mayer, L. Graser, H. Gupta, E. Saurez, and U. Ramachandran. "Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures". In: *2017 IEEE Fog World Congress (FWC)*. IEEE. 2017, pp. 1–6.

[130] I. Behnke, L. Thamsen, and O. Kao. "Héctor: A Framework for Testing IoT Applications Across Heterogeneous Edge and Cloud Testbeds". In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*. 2019, pp. 15–20.

[131] J. Hasenburg, M. Grambow, E. Grünewald, S. Huk, and D. Bermbach. "MockFog: Emulating fog computing infrastructure in the cloud". In: *2019 IEEE International Conference on Fog Computing (ICFC)*. IEEE. 2019, pp. 144–152.

[132] J. Hasenburg. *MockFog Inrastructure as Code*. https://github.com/OpenFogStack/MockFog-IaC. Accessed: 2020-07-24.

[133] L. Graser. *EmuFog on GitHub*. https://github.com/emufog/emufog. Accessed: 2020-07-24.

[134] A. Forster and A. L. Murphy. "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning". In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE. 2007, pp. 371–376.

[135] R. Arroyo-Valles, R. Alaiz-Rodriguez, A. Guerrero-Curieses, and J. Cid-Sueiro. "Q-probabilistic routing in wireless sensor networks". In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE. 2007, pp. 1–6.

[136] T. Hu and Y. Fei. "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks". In: *IEEE Transactions on Mobile Computing* 9.6 (2010), pp. 796–809.

[137] A. A. Bhorkar, M. Naghshvar, T. Javidi, and B. D. Rao. "Adaptive opportunistic routing for wireless ad hoc networks". In: *IEEE/ACM Transactions On Networking* 20.1 (2011), pp. 243–256.

[138] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. "Optimized IoT service placement in the fog". In: *Service Oriented Computing and Applications* 11.4 (2017), pp. 427–443.

[139] A. Medina, A. Lakhina, I. Matta, and J. Byers. "BRITE: An approach to universal topology generation". In: *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE. 2001, pp. 346–353.

[140] H. Gupta. *iFogSim on GitHub*. https://github.com/Cloudslab/iFogSim. Accessed: 2020-05-02.

[141] A. A. A. Ateya, A. Muthanna, R. Kirichek, M. Hammoudeh, and A. Koucheryavy. "Energy-and latency-aware hybrid offloading algorithm for UAVs". In: *IEEE Access* 7 (2019), pp. 37587–37600.

Chapter 3

Scalable design & dimensioning of fog-computing infrastructure to support latency-sensitive IoT applications

Abstract The Fog computing paradigm has appeared as a geo-distributed response to a growing focus on latency-sensitive Internet of Things (IoT) applications and the long-delay that may be provided by Cloud Data Centres. Although many researchers have investigated how IoT can interact with a Fog, very few have tackled the question of how to construct a Fog infrastructure for expected IoT traffic. This paper addresses the design and dimensioning of a Fog infrastructure via a Mixed-Integer Linear Program (MILP) to construct a physical Fog network design by mapping IoT virtual networks to dimensioned Fog nodes. Due to the exponential nature of this MILP formulation, we also propose a Column Generation model with near-optimal results at a significantly reduced design and dimensioning cost. Numerical results show the viability of the Column Generation method in its proximity to the optimal solution and in its reasonable solution time.

This chapter introduces fog design & dimensioning for stationary fog nodes to support IoT traffic over congested networks. This chapter has been published in IEEE Internet of Things [39].

Scalable Design and Dimensioning of Fog-Computing Infrastructure to Support Latency Sensitive IoT Applications

Ismael Martinez, Abdallah Jarray, and Abdelhakim Senhaji Hafid

Abstract—**The Fog computing paradigm has appeared as a geo-distributed response to a growing focus on latency-sensitive Internet of Things (IoT) applications and the long-delay that may be provided by Cloud Data Centres. Although many researchers have investigated how IoT can interact with a Fog, very few have tackled the question of how to construct a Fog infrastructure for expected IoT traffic. This paper addresses the design and dimensioning of a Fog infrastructure via a Mixed-Integer Linear Program (MILP) to construct a physical Fog network design by mapping IoT virtual networks to dimensioned Fog nodes. Due to the exponential nature of this MILP formulation, we also propose a Column Generation model with near-optimal results at a significantly reduced design and dimensioning cost. Numerical results show the viability of the Column Generation method in its proximity to the optimal solution and in its reasonable solution time.**

Index terms— Fog computing, Mixed-Integer Linear Programming, Column Generation, Design and Dimensioning, Internet of Things (IoT)

I. Introduction

As Internet-of-Things (IoT) devices become more prominent in all facets of industry, the number of IoT applications requiring real-time processing is also increasing. Heart monitors and vehicular traffic status sharing are examples of IoT applications that are increasing in number and use, and require near-immediate data transmission, processing and response by the Cloud. Processing through Cloud Data Centres result in unacceptably high latency for IoT application responses due to the centralized and distant nature of these Data Centres. A report by Cisco emphasizes that existing Cloud Data Centres are not designed to meet the current volume and variety of IoT data and requests. Meanwhile, the number of IoT devices are only expected to grow up to 50 billion by the year 2020 [1]; this means that processing of IoT applications outside of the Cloud is crucial.

To combat high round-trip latency and increasing data volume to Cloud Data Centres, Fog computing has been proposed as highly virtualized Micro Data Centres on the network edge [2], allowing the Fog network to analyze and process the most time-sensitive data. Although typically having fewer resources than the Cloud [3], Fog nodes are decentralized and geographically distributed enabling IoT connectivity with minimal round-trip latency. Furthermore, the Fog network can process many IoT applications that would otherwise be processed in the Cloud, reducing the volume of data reaching the Cloud. A wide-reaching Fog network could therefore satisfy any IoT process regardless of location.

I. Martinez is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ismael.martinez@umontreal.ca).

A. Jarray is with the School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada (e-mail: ajarray@uottawa. ca).

A. S. Hafid is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ahafid@iro.umontreal.ca).

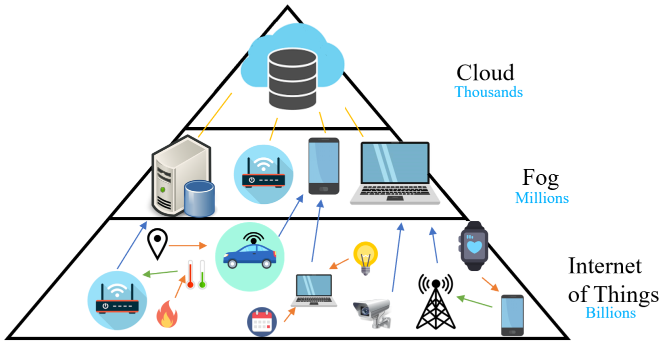


Fig. 1: Fog-DC infrastructure to process IoT requests.

The increase in IoT devices is a direct result of an acknowledgement of value, proposal, and deployment of IoT applications across many domains including health care [4] and Smart Cities [4]–[6]. The low latency, scalability and geo-distributed nature of Fog have made it integral to the success of these IoT applications [7]. The future growth and adoption of 5G radio access networks further facilitate the viability and implementation of Fog networks, and widen the scope of devices that can partake and aid in IoT-Fog communication [8].

IoT applications proposed to take advantage of Fog include energy management in residential domains [9], fire detection & fire-fighting [10], video streaming [11], and video surveillance [12]. There have been proposals for health care applications using Fog with existing wearable technologies [13], as well as medical-specific wearables to provide real-time assisted living services in hospitals [14] or to help diagnose heart disease [15].

Most existing contributions [13], [16]–[22] are concerned with Fog resource allocation to support the requirements of IoT applications; however, all these contributions assume an existing Fog computing infrastructure. Other State of the Art research in Fog computing focus on conceptual Fog infrastructure, and Fog interaction with IoT and Cloud. There has been only one study [23] that investigated the design of a Fog infrastructure; however, the proposed design was specific to Internet of Vehicle (IoV) applications. Indeed, there are no studies on the design and dimensioning of the Fog network for the support of the general IoT paradigm. We intend to provide a scalable and efficient approach to Fog design and dimensioning that can be feasibly implemented and deployed.

We consider a geographical region where no Fog infrastructure currently exists, and where there is an expected increase in IoT traffic. We define the *design* of the Fog infrastructure as the selection of Fog node locations from a set of candidate locations, and define the *dimensioning* of each Fog node as determining the amount of CPU, Memory and Storage resources to be installed. Designing and Dimensioning will therefore yield a full blueprint necessary for construction of a Fog infrastructure.

To the best of our knowledge, this paper is the first work to propose a design and dimensioning scheme of the general Fog infrastructure for IoT applications. Our approach is based on existing or predicted IoT traffic, with the goal to design a Fog infrastructure to support the processing of upcoming IoT requests. Furthermore, our approach is extensible, allowing for new Fog nodes to be optimally added to a current infrastructure in the event of an increase of IoT traffic.

We consider an infrastructure of the combined Fog and Cloud paradigms known as *Fog-DC* (Fig. 1), and optimize the design, dimensioning and IoT-resource allocation to the Fog-DC infrastructure. To optimize the cost of the Fog-DC design and dimensioning, two approaches are presented: an exact approach using a Mixed Integer Linear Programming (MILP) method called *Fog-DC-MILP*, and a heurisitc solution using Column Generation called *Fog-DC-CG*.

This paper outlines the first study of the design and dimensioning of Fog computing for IoT applications. Our main contributions are as follows:

1) We perform an analysis of model considerations such as transmission delay from IoT applications and network congestion towards Fog-DC that impact the design and dimensioning of the Fog-infrastructure.
2) We propose an exact optimization approach for Fog-DC design and dimensioning via one-shot MILP model (Fog-DC-MILP) that minimizes the resource and network mapping cost between a fixed IoT set and the constructed Fog-infrastructure.
3) We propose a scalable heuristic Column Generation approach (Fog-DC-CG) to overcome computation complexity of Fog-DC-MILP. Numerical results compare the efficiency of these two models with standard heuristics such as a Greedy Algorithm and a matching-based model.

Devising a scalable method to constructing an optimal Fog-infrastructure paves the way for implementation of IoT-Fog-Cloud communication. Given the numerous studies into the interaction between IoT, Fog and Cloud reviewed in the next section, physical Fog networks will allow for extended and practical research into the efficiency of Fog-computing.

The rest of this paper is organized as follows. Section II presents related work. Section III defines Fog-DC. Section IV presents the impact of IoT traffic, network congestion and routing delays on the Fog-DC model. Section V defines the MILP model. Section VI presents the Column Generation model. Section VII defines two benchmark heuristic models to compare against our proposed model. Section VIII evaluates and compares the proposed solutions. Section IX concludes the paper and presents future work.

II. Related Work

Conceptual research in Fog infrastructure has studied how Fog could be set-up to interact with IoT devices, and many proposals supporting IoT applications using an existing Fog infrastructure have been brought forth. Across different studies, we use IoT *requests* and *applications* interchangeably, and use *tasks* and *modules* interchangeably as components of an IoT application.

Three IoV applications use clusters of slow moving or parked cars as the Fog itself; Sookhak et al. [24] propose incentives for participating such as free parking, free Wi-Fi or free shopping vouchers, Hou et al. [25] show how non-smart cars may be upgraded with hardware and/or software in order to take part, and Zhou et al. [26] allocates Vehicular Fog resources to users in an information incomplete environment using preference matching algorithms. More generally, Chang et al. introduced the concept of *Consumer as a Provider* (CaaP), a platform to allow user devices such as phones and modems to act as Fog nodes available for public use [27]. With a large enough user base, the CaaP Fog network can cover vast continuous areas. Given an existing Fog infrastructure, Souza et al. [17] propose to virtually cluster Fog nodes for seamless data-sharing within clusters, and data-transfers between clusters. Zhang et al. [28] propose CFC-IoV as an IoV application whereby road side Fog nodes handover data between each other for moving vehicles.

Taneja et al. [29] note that each IoT request is in fact comprised of multiple *modules*, each comprised of a single sensor or actuator. Using heuristic search methods, Xia et al. [30] assigns IoT application modules to Fog devices. The module assignments reserve resources for the IoT application for all future requests until relinquished. Aazam et al. [31] developed a pricing model for module assignments based on the probability that an IoT application will relinquish its hold of Fog resources. By clustering IoT devices together that run the same service, and mapping those services to Fog nodes, Yousefpour et al. [32] are able to extend the scope of the IoT devices which can use each assigned modules.

Frameworks over a combination of IoT, Fog, and Cloud layers can facilitate resource allocation deployment. *FogBus* [33] is a framework where IoT requests first arrive to Broker nodes to be assessed and forwarded to Fog or Cloud Data Centres. FogBus also separates Fog nodes into computation and storage devices; these extra layers may decrease provisioning efficiency and functionality. The *Foggy* [34] framework is built using only a single Fog layer. Both frameworks require extra software installed in each Fog node to achieve inter-communication of available resources and job lists.

Resource provisioning schemes aim to find an acceptable mapping between IoT resources and Fog nodes. The objective of these schemes vary from minimizing round-trip latency [16], [30], maximizing Fog utility which inherently minimizes Cloud utility [3], [19], or minimizing the resource provisioning cost [13], [35]. Uniquely, Ni et al. [36] focus on minimizing Fog credibility score, and Do et al. [11] propose a model that jointly maximizes Fog utility and minimizes the generated carbon footprint. Three separate models are proposed by Zhang et al. with independent objectives: to minimize energy and delay costs of edge network resource allocation [37], to minimize IoT request rejection from Fog nodes [28], and to minimize the stochastic delay cost associated with edge computing [38].

Naas et al. [16] propose an Integer Programming formulation, and a resource provisioning heuristic based on geographical zoning that was evaluated using the simulation toolkit iFogSim [39]. Donasollo et al. [35] solve an Integer Program using a Divide & Conquer approach while minimizing resource allocation cost. Intharawijitr et al. [21] propose the use of 5G mobile networks with Fog to minimize the number of rejected IoT requests; an MILP was formulated and solved via a greedy allocation policy. Souza et al. [17] also formulates an MILP model over a Fog-Cloud infrastructure that was solved directly via Gurobi Optimizer for a limited instance size. Salaht et al. [40] use a Constraint Programming approach to satisfy all QoS requirements of IoT requests for a quicker solution; however, this can result in a large variety of provisioning costs to a user. Skarlat et al. [20] proposes a Mixed Non-Linear Programming model that is solved using a genetic algorithm.

Most resource provisioning approaches use the Fog alongside the Cloud to process IoT applications the Fog cannot [16], [17], [20], while [21] establishes that IoT requests not processed by the Fog are not processed at all. The rejection of IoT requests occurs when all Fog nodes that could satisfy the IoT latency threshold do not have enough available resources. To avoid this rejection, either a larger Fog infrastructure should be implemented, or a *network sink* such as the Cloud should be used to receive and process all IoT applications unable to be processed by the Fog infrastructure.

The core limitation of the reviewed research is the assumption that a Fog infrastructure exists. To the best of our knowledge, there is only one work that designs a Fog infrastructure to support IoV applications [23].

Yu et al. [23] propose a Fog deployment and dimensioning scheme for IoV applications, whereby vehicles connect to Road-Side Units (RSUs), which connect to Fog nodes in order to receive and provide real-time traffic conditions on all covered areas. Additionally, the optimal location of Gateways to access the Cloud are also found. The locations and dimensioning of RSUs, Fog and Gateway devices are optimally found from a set of candidate locations and dimensioning configurations in an arrangement that minimizes infrastructure cost. Regarding vehicular traffic, the model assumes a known static set of vehicle resources accessing the network across different regions, which may not be true in practice. Although having a set of candidate locations for Fog nodes is practical, the model also assumes a finite candidate set of dimensioning configurations; the solution to this model is thus dependent on the completeness of such a set. Finding the optimal placement of RSUs and Gateways increase the complexity of the model, while the inclusion of RSUs also restricts the application of this model to IoV.

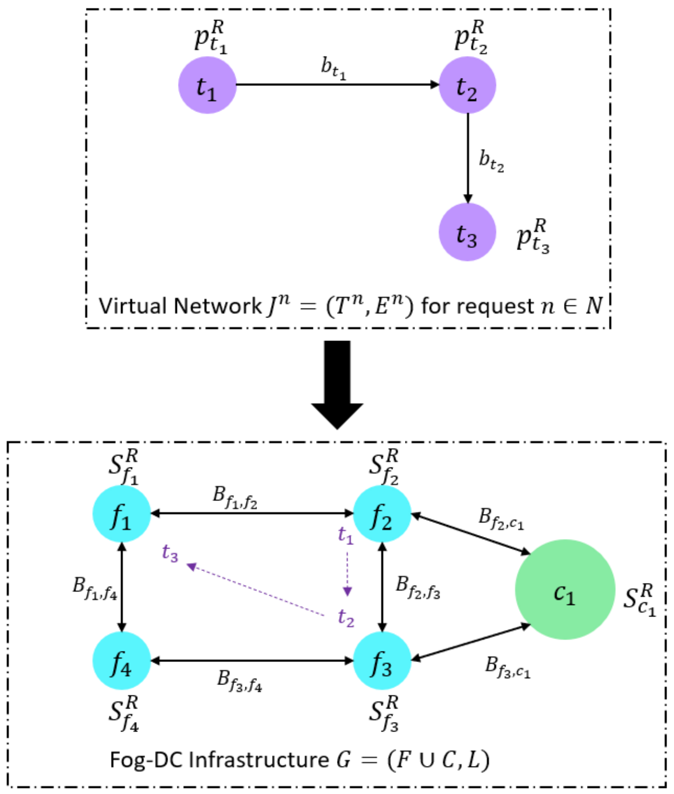


Fig. 2: Mapping of an IoT request virtual network $I_n = (T_n, E_n)$ to Fog-DC infrastructure $G = (F \cup C, L)$.

In this paper, we propose a Fog design and dimensioning scheme to support a general heterogeneous IoT population for a predicted set of IoT traffic. In place of using an exhaustive finite set of resource dimensions from which we allocate to each Fog node, we define the dimensions of each Fog node as continuous below its maximum resource capacity. Furthermore, by assuming any Fog has network access we acknowledge that any Fog node could access the Cloud, thus removing the need to place Gateways.

III. Fog-DC

A. *IoT Traffic*

We consider the total amount of expected IoT device tasks[1] for a given area made up of sensors and actuators, where each IoT request is comprised of one or more of these devices. We use the these expected number of tasks as well as the frequency of requests to formulate our predicted IoT traffic.

We define a set K of unique request classes[2], and use this set to build our expected IoT request set N. Each request class $k \in K$ defines a set of IoT applications with identical

[1]These estimates may come from estimations of the planned manufacturing and business expansion of the given area.

[2]Multiple requests executed from the same set of devices, with the same transmitted data format and expected response format at different times are considered to follow the same unique request class.

data transmission and response format, and number of request tasks q_k; furthermore, we assume data uploads from requests with class k follow a Poisson process with arrival rate λ_k [22].

Suppose we want our Fog-DC design to be built to handle the incoming traffic the majority of the time; we define $\rho \in [0.5, 1)$ to be the percentile of IoT traffic we wish to cover. For each IoT request class $k \in K$ with arrival rate λ_k, we define the ρ-percentile as

$$x_k^\rho = \min\{x_k \in \mathbb{Z}^+ : \mathcal{P}(x_k; \lambda_k) \geq \rho\}, \quad k \in K, \tag{1}$$

where $\mathcal{P}(x; \lambda)$ is the cumulative distribution function of the Poisson distribution with parameter λ. Let k_m represent the mth request of class k within a larger set. We define the ρ-percentile set of IoT requests as

$$N_\rho = \bigcup_{k \in K} \bigcup_{m=1}^{x_k^\rho} k_m, \tag{2}$$

where $\bigcup$ defines the union of sets. By construction of N_ρ in (2), we observe a mapping between N_ρ and K such that for each request $n \in N_\rho$, there exists a class $k \in K$ associated with n. We define this mapping as $M_{\text{K}} : N_\rho \mapsto K$. If $M_{\text{K}}(n) = k$, then the number of tasks in request n is q_k. Moving forwards, for ρ-percentile of expected IoT traffic, we define $N = N_\rho$ to be the set of IoT requests. Since $\mathcal{P}(x; \lambda)$ is monotonically increasing for fixed λ, larger values of ρ yield a larger set N.

B. IoT Virtual Network

Each of the IoT requests is divisible into a set of interdependent atomic tasks modeled as a weighted directed virtual network graph, represented by a Task Dependency Graph (TDG) $I_n = (T_n, E_n)$ (Fig. 2), for request $n \in N$; T_n denotes the set of tasks in request n, and E_n denotes the set of directed virtual networking links between tasks.

For a request $n \in N$, each task $t \in T_n$ has a set of VM Cloud computing requirements: (a) CPU capacity $p_t^{\texttt{CPU}}$, (b) memory processing requirement $p_t^{\texttt{MEM}}$, (c) storage requirement $p_t^{\texttt{STR}}$, and (d) processing order o_t with respect to any other $t' \in T_n$. For simplicitly, we define the set $R = \{\texttt{CPU}, \texttt{MEM}, \texttt{STR}\}$ to be the set of VM resource requirements. This processing order o_t is used for the construction of the IoT TDG I_n.

C. Fog-DC Physical Infrastructure

The Fog-DC physical infrastructure is represented by a directed graph $G = (W, L)$ (Fig. 2) where $W = F \cup C$ is the set of potential Fog node locations F and available Cloud Data Centre sites C in the vicinity of the designed infrastructure, and L is the set of wired/wireless bidirectional links connecting Fog-DC nodes.

Each Fog-DC node $w \in W$ has a maximum available resource limit of S_w^r, $r \in R$. Between any two Fog-DC nodes $w, w' \in W$, the link $l_{(w,w')} \in L$ has a maximum bandwidth capacity of $B_{(w,w')} > 0$.

The set of Fog-DC nodes $W = F \cup C$ contains both Fog node candidates and fixed Cloud nodes. This leverages the low-latency benefits of Fog nodes on the network edge, and the large memory and storage capabilities of the Cloud. The result is a designed and dimensioned architecture that can satisfy low-latency responses from IoT with Fog, and pass long-term storage and latency insensitive application processing to the Cloud.

Altogether, we have $|N|$ virtual networks – one for each IoT request – and one physical network of Fog nodes. For each IoT virtual network $I_n, n \in N$, we want to establish a network mapping to the Fog-DC infrastructure G. A feasible mapping of all IoT tasks to Fog-DC nodes must adhere to the resource and bandwidth capacities of the Fog-DC infrastructure.

The mapping of each IoT request I_n (Fig. 2) can be divided into hosting and network mapping. Each IoT hosting node $t \in T_n$ belonging to an IoT request $n \in N$ is mapped to a distinguished Fog-DC node $w \in W$ by mapping $M_{\text{W}} : T_n \mapsto W$.

We define the set of all possible paths between any two nodes in W as Π, and the set of all possible paths between two specific nodes $w, w' \in W$ as $\Pi_{(w,w')}$. Each virtual IoT link $e = (t, t'), e \in E_n$ belonging to an IoT request $n \in N$ is mapped to an Fog-DC path $\pi \in \Pi_{(w,w')} \subseteq \Pi$ via $M_{\text{E}} : E_n \mapsto \Pi_{(w,w')}$, where $M_{\text{W}}(t) = w$ and $M_{\text{W}}(t') = w'$. Note, if $w = w'$, then $\Pi_{(w,w')} = \varnothing$ since no path is needed.

IV. Model Considerations

A. Reachable Servers

The large selling point of a Fog infrastructure closer to an IoT device is reduced latency which benefits applications needing real-time or near real-time responses. Consider the IoT device from which a task $t \in T_n$, $n \in N$ is executed; for simplicity, we refer to this device as IoT device t. Suppose IoT device t has a range[3] γ_t, and a distance $d_{t,w}$ between an IoT device t and a Fog node $f \in F$. If $d_{t,f} \leq \gamma_t$, we consider this Fog node *reachable* from t, and if $d_{t,f} > \gamma_t$,then f can be accessed by t by one of many available access points a within the task's radius of communication. Similarly, either a Fog node f or an access point a can act as a gateway to reach any Cloud Data Centres. This communication is visually represented in Fig. 3.

B. Network Congestion

In order to correctly model the latency given to IoT-Fog mapping, we must consider the effect IoT traffic has on physical link congestion. For a request $n \in N$ such that $M_{\text{K}}(n) = k$, let $\{X_n(\tau), \tau \geq 0\}$ be the volume of transmitted IoT data by time τ following a Poisson Process with rate λ_k per unit time, i.e. $\mathbb{E}[X_n] = \lambda_k, \forall n \in N, M_{\text{K}}(n) = k$. The total rate of data entering Fog-DC is the sum of these individual request amounts; therefore, the overall arrival rate is

$$\sum_{n \in N} \lambda_{M_{\text{K}}(n)} = \lambda_\rho. \tag{3}$$

By construction of N in (2), an increase in ρ to ρ' will yield an increased arrival rate $\lambda'_{M_{\text{K}}(n)}, \forall n \in N$, hence an increased overall arrival rate $\lambda_{\rho'}$.

[3]A survey on IoT found the range of most devices were from 10 to 100 meters, depending on the communication standard and technology used [41].

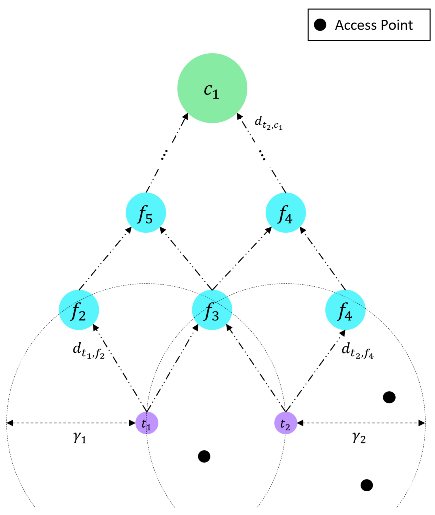


Fig. 3: Each task can directly communicate with any Fog node within its range of communication γ. Fogs can also be accessed via an internet access point at a greater latency cost. Every Fog node and access point can act as a gateway to Cloud devices.

Within the Fog, we define the *outermost* nodes to be the nodes that fall within the radius of communication γ_t of at least one IoT device t; these are also called the *edge* nodes. We define the *innermost* nodes to be the Fog nodes that link directly to a Cloud Data Centre. All nodes in between are referred to as *core* Fog nodes.

Regardless of where data from task t is processed, we assume all data reaches the Cloud at some point either for processing or long-term storage. In Fig. 4 we have divided our system into 3 layers – IoT, Fog and Cloud. The amount of data that enters the Fog follows a Poisson Distribution with rate λ_ρ and expects a response for each datapoint; we assume the amount of data that eventually enters the Cloud is equivalent. In other words, for every data point transmitted by IoT device t, the data's lifecycle includes being processed by a node either in the Cloud or the Fog network with a response being returned, and long-term storage being propagated to the Cloud.

Since we assume all data must pass through the Cloud, we recognize substantial link congestion entering the Cloud; however, having some data processing occur in the Fog results in reduced link congestion returning from the Cloud. Similarly, there may be higher link congestion as we move towards the innermost nodes, but the amount of link congestion from processing responses are reduced as we process more tasks in the outermost nodes.

Congestion on a single link can be modelled as an $M/M/1$ queue, where requests are processed and released at an exponentially distributed rate μ_w by $w \in W$. We define p_w to be the probability that a Fog-DC node $w \in W$ receives the executed IoT request such that $\sum_{w\in W} p_w = 1$; this probability is based on available resources of w and distance from the IoT devices. We also define the expected number of IoT requests received by w as $p_w\lambda_\rho$. Using the IoT data arrival and service rates, as well as Little's Law [42], we determine the average wait time to be processed by $w \in W$, denoted $\bar{\eta}_w$ to be

$$\bar{\eta}_{w;\rho} = \frac{p_w\lambda_\rho}{\mu_w(\mu_w - p_w\lambda_\rho)}. \tag{4}$$

For a fixed u_w and p_w, $\bar{\eta}_w$ is monotonically increasing with $\lambda_\rho > 0$. By (2) and (3), λ_ρ is monotonically increasing with the percentile ρ; therefore, an increased percentile ρ leads to a larger average wait time.

Let $L_w = \{l \in L \mid l = (w_0, w) \in L\} \cup \{(t, w) \mid M_{\mathrm{W}}(t) = w\}$ be the set of links feeding into w, and let $H_{w;\rho}$ denote the random variable of waiting time before being processed by node $w \in W$ for a network accepting the ρ-percentile of IoT traffic. We know $\mathbb{E}[H_{w;\rho}] = \bar{\eta}_{w;\rho}$ by definition. By [42], the cumulative distribution $G_{w;\rho}(x)$ of $H_{w;\rho}$ is

$$G_{w;\rho}(t) = \mathbb{P}[H_{w;\rho} \le t] = 1 - \frac{p_w\lambda_\rho}{\mu_w} e^{-(\mu_w - p_w\lambda_\rho)t}. \tag{5}$$

We denote this wait time $H_{w;\rho}$ to represent our *congestion factor* for a link (w_0, w), $w \in L_w$, and ρ-percentile IoT traffic. For any response from w to t, we define the service rate $\mu_t = 0$; therefore, we define $H_{t;\rho} = 0$, $t \in T_n, n \in N$.

Let π_W be the set of all nodes $w \in W \cup T_n, n \in N$ in a path $\pi \backslash \{w_0\}$ $\pi \in \Pi_{(w_0,w)}$. To estimate congestion over a path from w_0 to w', we simulate the maximum waiting time over all paths $\pi \in \Pi_{(w_0,w)}$ which gives a worst-case congestion per simulation, and calculate the mean and variance. By Central Limit Theorem, [43], we can approximate the congestion factor over the any path by a normal distribution for a large number of m simulations.

Algorithm 1 Simulate congestion over a path w_0 to w

```
Result: Estimated congestion distribution $H_{(w_0,w);\rho}$ for $\rho$-percentile IoT traffic.
Initialize $\rho \in [0.5, 1)$;
for $i$ = 1 to $m$ do
    for $w \in W \cup T_n, n \in N$ do
        $\eta_w \leftarrow H_{w;\rho}$    // Sample
    end
    $\tilde{\eta}_{(w_0,w),i} \leftarrow \max\{\sum_{w\in\pi_W} \eta_{w;\rho} : \pi \in \Pi_{(w_0,w)}\}$
end
$\bar{\eta}_{(w_0,w)} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \tilde{\eta}_{(w_0,w),i}$
$\hat{\sigma}^2_{w_0,w} \leftarrow \frac{1}{m-1}\sum_{i=1}^{m}(\tilde{\eta}_{(w_0,w),i} - \bar{\eta}_{(w_0,w)})^2$
$H_{(w_0,w);\rho} \sim \mathcal{N}(\bar{\eta}_{(w_0,w)}, \hat{\sigma}^2_{w_0,w})$ .
```

Note, a path $\Pi_{(w_0,w)}$ has at most two elements in $T_n, n \in N$ at the beginning and/or end of the sequence. Let $G_{(w_0,w);\rho}(t)$ represent the cumulative distribution function of $H_{(w_0,w);\rho}$.

Once an IoT request is processed by a Fog node and a response is returned to the IoT device, the request data is also sent to the Cloud for long-term storage; however, since no further processing of the request is needed, the link congestion associated with this data instance is negligible. From this, we infer that congestion towards the Cloud is alleviated as the probability that a request is processed within the Fog network instead of the Cloud increases.

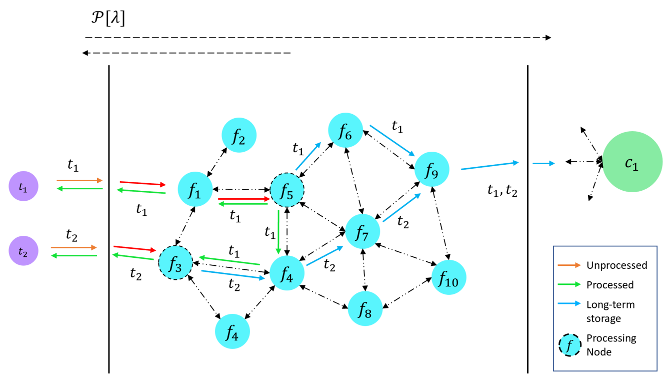


Fig. 4: The Fog-DC infrastructure divided into IoT, Fog, and Cloud to illustrate how congestion may occur. Shown is the lifecycle of a single request with tasks t_1 and t_2; a response to t_1 is required from Fog f_5, and a response to t_2 is required from t_3. Once processed, data is also propagated to the Cloud for long-term storage.

We make the simplistic assumption that a Fog-DC node w can only process one IoT request at a time, which is not true in practice. Furthermore, our current analysis of network congestion is based on placing a Fog node f in every candidate location in F; in reality, only a handful of these locations will be used, raising the congestion within the Fog network. To determine a more accurate estimate of how much congestion is present surrounding different nodes in the network, a simulation is necessary provided a network architecture and expected IoT traffic; a simulation framework such as iFogSim [39] could be used, which we propose to do in a future work.

C. Transmission Delay

Each task t has a transmission file size s_t^+ and a bandwidth requirement[4] b_t that remain consistent throughout transmission from t to t' where $o_t = o_{t'} - 1,\ t, t' \in T_n,\ n \in N$. Wang et al. [45] found a positive correlation between the number of hops and the latency of internet communication between two nodes under 1000 kilometres in distance; therefore, we model transmission delay using hops.

For Fog node $f \in F$, let $G_{f;\rho}^{-1}(\cdot)$ denote the inverse cumulative distribution function of $H_{f;\rho}$, and $G_{(t,w);\rho}^{-1}(\cdot)$ denote the inverse cumulative distribution function of $H_{(t,w);\rho}$. For $U \sim Uniform(0,1)$, we can represent a random sample of congestion over a path as $H_{f;\rho} = G_{f;\rho}^{-1}(U)$ and $H_{(t,w);\rho} = G_{(t,w);\rho}^{-1}(U)$. This congestion can also be interpreted as queuing delay or wait time of a package over the specified link. Let $\omega \in (0,1)$ represents the percentile of estimated congestion for a random path; higher values of ω yield a worst-case estimate of transmission delay.

For a Fog-DC node $w \in W$, the transmission delay is dependent on the transmission bandwidth, the data file size, an upper bound on the number of hops from t to w denoted $h_{t,w}$, and any network congestion found along the way. Based on these factors, we devise an approximation to the communication delay between a task t and a Fog-DC node w denote $\nu_{t,w}^{\rho,\omega}$.

For a Fog node $f \in F$ such that $M_{\text{W}}(t) = f$ and $d_{t,f} \leq \gamma_t$, and ρ-percentile IoT traffic, we get the transmission latency as

$$\nu_{t,f}^{\rho,\omega} = \frac{s_t^+}{b_t} + G_{f;\rho}^{-1}(\omega). \tag{6}$$

If a Fog-DC node $w \in W$ is not within the communication range γ_t of a task t, the expected transmission delay becomes

$$\nu_{t,w}^{\rho,\omega} = h_{t,w} \cdot \frac{s_t^+}{b_t} + G_{(t,w);\rho}^{-1}(\omega), \tag{7}$$

where we only consider one hop of congestion entering the Fog-DC infrastructure.

We assume the bandwidth of a task remains constant throughout its transmission and response, a response file size s_t^- after processing, and a ω-percentile of congestion on any path. Then, for a task mapping $M_{\text{W}}(t) = w$, we have

$$\begin{aligned} \nu_{t,w}^{\rho,\omega} &= h_{t,w} \cdot \frac{s_t^+}{b_t} + G_{(t,w);\rho}^{-1}(\omega) \\ \nu_{w,t}^{\rho,\omega} &= h_{w,t} \cdot \frac{s_t^-}{b_t} + G_{(w,t);\rho}^{-1}(\omega) \end{aligned} \tag{8}$$

for $h_{t,w} = h_{w,t}$.

D. Physical Network Delay

Data transmitted by t to w is a package of size s_t^+ with response package size s_t^-. For a pair of ordered and successive tasks t, t' and task mappings $M_{\text{W}}(t) = w$ and $M_{\text{W}}(t') = w'$ the network delay is dependent on the task bandwidth b_t, the response file size s_t^-, the ρ-percentile of IoT traffic, and the ω-percentile of resulting congestion. We define the network delay from a task t as

$$\phi_t^{\rho,\omega}(w, w') = h_{w,w'} \cdot \left(\frac{s_t^-}{b_t} + G_{(w,w');\rho}^{-1}(\omega) \right), \tag{9}$$

where $h_{w,w'}$ is an upper bound on the number of hops between w and w', and $\omega \in (0,1)$ is the percentile of congestion expected on a random path $\pi \in \Pi_{(w,w')}$.

For a single request $\{n \in N \mid M_{\text{K}}(n) = k\}$, we order the request tasks $t_1, ..., t_{q_k}$, each with differing file sizes and bandwidth requirements to the next task. For $M_{\text{W}}(t_i) = w_i$, the total physical network delay for request $n \in N$ is

$$\phi_n^{\rho,\omega}(w_1, w_{q_k}) = \sum_{i=1}^{q_k - 1} \phi_{t_i}^{\rho,\omega}(w_i, w_{i+1}). \tag{10}$$

When modelling a large network, these approximate calculations of physical network delay between each task may become too large. We invoke a worst-case estimate of physical network delay to our model to provide model scalability.

Consider a pair of Fog-DC nodes $w, w' \in W$, and let $\pi \in \Pi_{(w,w')}$ be the set of all possible paths between these two nodes. If we assume data can take any path between these

[4]The most common IoT transmission standard is IEEE 802.15.4 [41], which for a common frequency of 2.4 GHz transmits at 250kbps, with this rate decreasing with lower transmit frequencies [44], giving us an idea of the range of transmission bandwidth b_t.

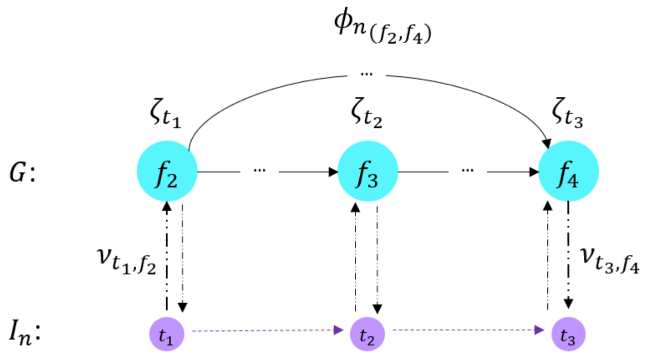


Fig. 5: The end-to-end delay for a sequence of tasks is the sum of transmission times, processing times and routing time.

nodes, we define π^* to be the critical path (longest path), and h_{π^*} to be the total number of hops taken to traverse this path. When we are considering the routing time from the first task to the last task, we assume the worst case scenario regardless of whether there exists shorter routing through the other tasks. We also define

$$r_n = \max_{i=1,\dots,q_k} \left\{ \frac{s^-_{t_i}}{b_{t_i}} \right\} \tag{11}$$

as the worst-case transmission rate. We then define

$$\tilde{\phi}^{\rho,\omega}_n(w_1, w_{q_k}) = h_{\pi^*}\left(r_n + G^{-1}_{(w_1,w_{q_k});\rho}(\omega)\right) \tag{12}$$

as the worst-case transmission rate with congestion percentile $\omega \in (0,1)$.

E. Total Delay

For a task t and mapping $M_W(t) = w$ we incur the transmission delays $\nu^{\rho,\omega}_{t,w}$ to transmit and $\nu^{\rho,\omega}_{w,t}$ to receive a response (8).

Once a task is transmitted to a Fog node, it must be processed before moving on. Each task t has a set computing time ζ_t which is specific to the set of resources needed for that task $\{p^r_t \mid r \in R\}$; as a result, it does not depend on the Fog node on which it is executed.

For a mapping $M_K(n) = k$, let the tasks of the request n be ordered $t_1, t_2, \dots, t_{q_k}$ such that $t < t'$ if $o_t < o_{t'}$. For a task mapping $M_W(t_i) = w_i$ for each $t_i \in T_n$, the end-to-end delay with ρ-percentile IoT traffic and ω-percentile congestion can be defined as

$$\nu^{\rho,\omega}_{t_1,w_1} + \tilde{\phi}^{\rho,\omega}_n(w_1, w_{q_k}) + \nu^{\rho,\omega}_{w_{q_k},t_{q_k}} + \sum_{i=1}^{q_k} \zeta_{t_i}. \tag{13}$$

Fig. 5 shows that each task is uploading data to their paired Fog-DC node, and may receive a response. In order for a Fog-DC node w to process its paired task t, it must both receive the task data from t as well as the processed data from the preceding Fog node. We assume all tasks $t \in T_n$ for a request $n \in N$ are transmitted synchronously, so there exists no delay between the start of each task. To this end, we make the assumption that for a task mapping pair $M_W(t_i) = w_i$ and $M_W(t_j) = w_j$, $o_{t_i} < o_{t_j}$ with ρ-percentile IoT traffic and ω-percentile congestion, we have

$$\nu^{\rho,\omega}_{t_i,w_i} + \zeta_{t_i} + \tilde{\phi}^{\rho,\omega}_n(w_i, w_j) \geq \nu^{\rho,\omega}_{t_j,w_j}. \tag{14}$$

This ensures w_j receives the task information from t_j prior to any data required from w_i, allowing w_j to begin on any processing required from the previous tasks immediately upon arrival.

Though here we assume a normal distribution of congestion and a worst-case routing delay, in practice we would use historical data on incoming IoT data to properly model a probability distribution describing routing delay patterns. In the event of no current IoT devices set up in the Fog designed area, we elect to use historical data of similar IoT devices from surrounding districts. This is left for future work.

V. MILP Model

Design and Dimensioning Approach

Design and dimensioning are ones of the most important aspects of Fog-DC management, since they are directly related to the cost and the QoS requirements of IoT computing services. Efficient design and dimensioning will have a positive impact on Fog-DC service provider's profitability. Resource allocation to the incoming IoT requests could be performed in a batch wise fashion. The size of batch can be modified depending on the considered topology and IoT traffic so as to ensure real time response to the requests. The Fog-DC design and dimensioning problem is in: (1) selecting the optimal location and dimensioning of Fog-DC sites, and (2) minimizing the cost of resources (e.g., computing and communication) while satisfying QoS requirements of IoT requests. These requirements include (1) bandwidth: transmission capacity between tasks of the IoT request , (2) latency: the time it takes to process the request and receive the response, (3) computing: computing capacity to process the tasks of the IoT request, (4) storage and memory capacity requirements of each IoT task, and, and (5) processing order of the tasks that composes the IoT request.

To evaluate the merits of the proposed Fog-DC design, dimensioning and resource allocation approach, we propose the following MILP based mathematical formulation we call Fog-DC-MILP. A virtual link $e \in E_n$ is defined by the pair $e = (t, t')$ such that $t, t' \in T_n, n \in N$, with $o_t = o_{t'} - 1$ meaning t' immediately follows t in the order of tasks. Each link $e \in E_n$ has a data transfer capacity requirement b_t between a pair of tasks t and t'. Furthermore, we invoke a partial latency requirement τ_t that the completion of all tasks up to and including t must satisfy.

Recall, for a request $I_n = (T_n, E_n)$, we define the task mapping as $M_W : T_n \mapsto W$ and the network mapping as $M_E : E_n \mapsto \Pi$. When an IoT request arrives, Fog-DC provider has to determine whether it is to accepted or rejected. This decision is largely based on the QoS requirements of the IoT request, the availability of Fog-DC resources, and the economic cost of accepting the request. The total cost of a

TABLE I: *Fog Infrastructure Network $G = (F, L)$*

Parameters	Description
F	Set of possible Fog locations.
C	Set of existing Cloud Data Centres.
W	Set of Fog-DC nodes; $W = F \cup C$.
L	Set of Fog-DC bidirectional links.
R	Type of resource that can be CPU, Memory or Storage. $R = \{\text{CPU}, \text{MEM}, \text{STR}\}$.
S_w^r	Maximum capacity of resource $r \in R$ in node $w \in W$.
B_l	Bandwidth capacity of physical link $l \in L$.
Y_{MAX}^r	Maximum number of Fog-DC locations that can receive resource $r \in R$.
c_l	Bandwidth unit cost of using substrate link $l \in L$.
c_w^r	Unit cost of using resource $r \in R$ in node $w \in W$.
u_f^r	Unit cost of setting up resource $r \in R$ in node $f \in F$.
D_f^r	Capital cost of setting up resource $r \in R$ in node $f \in F$.
Π	Set of paths in Fog-DC network.
$\Pi_{(w,w')}$	Set of paths between Fog-DC nodes w and w'.

request I_n is then represented as the combined cost of the task mapping, and the virtual link mapping as follows:

$$\text{COST}[I_n] = \text{COST}\left[M_{\text{W}}(T_n), M_{\text{E}}(E_n)\right]. \tag{15}$$

A. Decision Variables

1) Design and Dimensioning Variables: We recall that r is the type of Fog computing resource that takes values in the set $R = \{\text{CPU}, \text{MEM}, \text{STR}\}$. We define the float decision variable z_f^r to measure the amount of computing resources of type r required to be setup in Fog-DC node $f \in F$. An upper bound S_f^r will be setup to limit the available Fog computing resources of type r that can be setup in Fog-DC node f.

We define a binary decision variable that denotes whether Fog node $f \in F$ requires resources $r \in R$ to be setup; this is precisely whether z_f^r is greater than zero.

$$y_f^r = \begin{cases} 1, & \text{if } z_f^r > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

The variables z_c^r and y_c^r, $c \in C$ respectively represent the amount of resource r available in c, and whether this amount is strictly positive; these values are fixed as we cannot alter the resources provided in the Cloud.

TABLE II: *IoT virtual Network $I_n = (T_n, E_n)$*

Parameters	Description
N	Set of IoT requests.
I_n	Virtual network representing IoT request $n \in N$.
T_n	Set of tasks in an IoT request I_n.
E_n	Set of virtual directional links between tasks in an IoT request I_n.
R	Type of resource that can be CPU, Memory or Storage. $R = \{\text{CPU}, \text{MEM}, \text{STR}\}$.
p_t^r	Required capacity of resource of type $r \in R$ for task t.
o_t	Order number of task t in request T_n.
b_t	Transfer bandwidth between a task t up to the next ordered task.
s_t^+	File size of package uploaded by task t.
s_t^-	File size of package after processed by $w \in W$, either as a response to t or outgoing to the next task t'.
τ_t	The end-to-end delay threshold for the completion of the first task up to and including t.

2) IoT request mapping variables: To decide on the acceptance of an IoT request I_n, we need to define the following decision variables.

$$a_n = \begin{cases} 1, & \text{if an IoT request } n \in N \text{ is accepted to} \\ & \text{be processed by Fog-DC} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

$$x_w^t = \begin{cases} 1, & \text{if virtual node } t \in T_n,\ n \in N \text{ is assigned} \\ & \text{to Fog-DC node } w;\ M_{\text{W}}(t) = w \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$x_\pi^e = \begin{cases} 1, & \text{if virtual link } e \in E_n,\ n \in N \text{ is assigned} \\ & \text{to physical path } \pi \in \Pi;\ M_{\text{E}}(e) = \pi \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

The description and domain of fog-DC-MILP decision variables are summarized in Table IV.

B. Objective function

The objective of fog-DC-MILP is to minimize the total cost of fog design and dimensioning as defined by IoT request

TABLE III: *Model Considerations*

Variable	**Description**
K	Set of unique request classes.
$\mathcal{P}(x;\lambda)$	Poisson cumulative function with IoT traffic arrival rate λ.
ρ	Percentile of estimated IoT traffic volume entering network.
ω	Percentile of estimated congestion over any network path.
λ_k	Arrival rate of all requests with class $k \in K$.
λ_ρ	Sum of arrival rates over all classes using ρ-percentile IoT traffic.
p_w	Probability of a task t accessing Fog-DC node $w \in W$.
μ_w	Service rate of Fog-DC node $w \in W$.
$\bar{\eta}_{w;\rho}$	Mean waiting time/congestion of any link entering $w \in W$ with ρ-percentile IoT traffic.
$H_{w;\rho}$	Random variable representing congestion entering node $w \in W$.
$G_{w;\rho}(t)$	Cumulative Distribution Function of $H_{w;\rho}$.
$H_{(w,w');\rho}$	Random variable representing congestion over any path $\pi \in \Pi_{(w,w')}$ with ρ-percentile IoT traffic.
$G_{(w,w');\rho}(t)$	Cumulative Distribution Function of $H_{(w,w');\rho}$.
$\nu^{\rho,\omega}_{t,w}$	Upload transmission delay with ρ-percentile IoT traffic and ω-percentile congestion.
$\nu^{\rho,\omega}_{w,t}$	Download transmission delay with ρ-percentile IoT traffic and ω-percentile congestion.
$\phi^{\rho,\omega}_n(w_1, w_{q_k})$	Routing delay through intermediate nodes $w_1, ..., w_{q_k} \in W$ with ρ-percentile IoT traffic and ω-percentile congestion.
$\tilde{\phi}^{\rho,\omega}_n(w_1, w_{q_k})$	Worst-case routing delay with ρ-percentile IoT traffic and ω-percentile congestion.
ζ_t	Fixed processing time for task t.

mappings (15) as follows:

$$Z_{\text{MILP}} = \min \sum_{f\in F}\sum_{r\in R}\left(D^r_f \cdot y^r_f \;+\; u^r_f \cdot z^r_f\right) + \sum_{n\in N}\sum_{t\in T_n}\sum_{w\in W}\sum_{r\in R} c^r_w\, p^r_t\, x^t_w + \sum_{n\in N}\sum_{e\in E_n}\sum_{(w,w')\in W^2}\sum_{\pi\in\Pi_{(w,w')}}\sum_{l\in\pi} x^e_\pi\, c_l\, b_t \tag{20}$$

where D^r_f and u^r_f are the capital cost (Capex) and the unit cost respectively to setup computing resource of type $r \in R$ in Fog-DC node $f \in F$, and W^2 is the set of pairs $W \times W$.

We have divided the objective function (20) into three terms to express the optimization of different factors in Fog-DC-MILP: (a) The fixed and unit cost of resources placed in each Fog node $f \in F$ which represents the design and dimensioning cost, (b) the cost of allocating resource $r \in R$ for task $t \in T_n$ to Fog-DC nodes $w \in W$, and (c) the networking cost of assigning path $\pi \in \Pi_{(w,w')}$ to a virtual link $e = (t,t') \in E_n$, $n \in N$.

1) IoT resource allocation constraints: For the following constraints, we define Q_t as the set of possible Fog-DC sites $w \in W$ that can serve a task of an IoT request $t \in T_n$, and Q_e as the set of possible paths $\pi \in \Pi$ that can be assigned to $e \in E_n$. For a link $l \in L$, we define Π^l to be the set of paths that contain the link l, and $\Pi^l_{(w,w')}$ to be the set of paths from w to w' that contain link l.

$$\sum_{n\in N} a_n \geq |N| \cdot A \tag{21}$$

$$\sum_{w\in W} x^t_w = a_n\,;\; t \in T_n, n \in N \tag{22}$$

$$x^t_w = 0\;;\;\forall w \in W \setminus Q_t, \forall\, t \in T_n, n \in N \tag{23}$$

$$x^e_\pi = 0\;;\;\forall \pi \in \Pi \setminus Q_e, \forall\, e \in E_n, n \in N \tag{24}$$

$$x^t_w \cdot x^{t'}_{w'} = \sum_{\pi\in\Pi_{(w,w')}} x^e_\pi\,;\;(w,w') \in W^2,\quad e = (t,t') \in E_n, n \in N \tag{25}$$

$$\Delta^{t_i,\rho,\omega}_{(w_1,w_i)} \leq \tau_{t_i}\;;\; i \in \{1,...,|T_n|\},\, t_i \in T_n,\quad n \in N,\;\omega \in (0,1),\;(w_1,w_i) \in W^2 \tag{26}$$

$$\sum_{n\in N}\sum_{\substack{e\in E_n\\ e=(t,t')}}\sum_{(w,w')\in W^2}\sum_{\pi\in\Pi^l_{(w,w')}} b_t \cdot x^e_\pi \leq B_l\,;\; l \in L \tag{27}$$

$$\sum_{n\in N}\sum_{t\in T_n} x^t_w \cdot p^r_t \leq\, z^r_w\,;\; r \in R,\, w \in W. \tag{28}$$

Equation (21) expresses the acceptance ratio of the proposed Fog-DC design where parameter $A \in [0,1]$. Equations (22) and (23) express the selection of Fog-DC nodes to process IoT request tasks. Equations (24) and (25) ensure that only one valid embedding path is assigned for each virtual link. In (26), we define the partial delay Δ between the first task t_1 and the ith task t_i with transmission delay and congestion percentile $\omega \in (0,1)$ to be

$$\Delta^{t_i,\rho,\omega}_{(w_1,w_i)} = \nu^{\rho,\omega}_{t_1,w_1} \cdot x^{t_1}_{w_1} + \tilde{\phi}^{\rho,\omega}_{(w_1,w_i)} \cdot x^{t_1}_{w_1} \cdot x^{t_i}_{w_i} + \nu^{\rho,\omega}_{w_i,t_i} \cdot x^{t_i}_{w_i} + \sum_{m=1}^{i} \zeta_{t_m},\; t \in T_n,\; n \in N, \omega \in (0,1) \tag{29}$$

which expresses the QoS requirements for the mapping of virtual links to meet the latency threshold of tasks. Recall, ρ is the percentile of IoT traffic that both defines the set N and the congestion distribution, therefore it is set prior to modelization. We note that the term $\phi^{\rho,\omega}_{(w_1,w_i)} \cdot x^{t_1}_{w_1} \cdot x^{t_i}_{w_i}$ in (29)

TABLE IV: *Fog-DC-MILP Decision Variables*

Decision Variables	**Domain**	**Description**
z_f^r	$\mathbb{R}, \geq 0$	Dimension of resources $r \in R$ allocated to Fog node $f \in F$.
z_c^r	$\mathbb{R}, \geq 0$	Dimension of resources $r \in R$ allocated to Cloud node $c \in C$; `FIXED`.
y_f^r	{0,1}	Whether a resource $r \in R$ is present in Fog node $f \in F$.
y_c^r	{0,1}	Whether a resource $r \in R$ is present in Cloud node $c \in C$; `FIXED`.
a_n	{0,1}	Whether a request I_n is accepted by Fog-DC.
x_w^t	{0,1}	Whether task $t \in T_n$ is mapped to Fog-DC node $w \in W$.
x_π^e	{0,1}	Whether virtual link $e = (t, t') \in E_n$ is mapped to physical path $\pi \in \Pi$.

is quadratic, and we use the same linearization technique as in Section VI-C4. Equations (27) and (28) express respectively the bandwidth and resource (CPU, Memory and Storage) capacity of links and nodes.

2) Fog-DC Design and Dimensioning Constraints:

$$z_f^r \leq S_f^r \cdot y_f^r \; ; \; f \in F, \; r \in R \tag{30}$$

$$\sum_{f \in F} y_f^r \leq Y_{\text{MAX}}^r \; ; \; r \in R. \tag{31}$$

Equation (30) expresses that no resources r can be setup in a given Fog node f if it is not selected as an optimal location to setup a Fog node, and must not exceed the maximum resource amount for that Fog location denoted by S_f^r. Equation (31) expresses the maximum number of Fog-DC locations Y_{MAX}^r that can receive a resource r.

C. Extensibility of Fog-DC forumaltion

Suppose a Fog infrastructure already exists and we wish to extend it to support higher IoT traffic. We define the current set of dimensioned Fog nodes as F'. Into the current formulation, we integrate $y_f^r = 1$ fixed and $z_f^r \in [\bar{z}_f^r, S_f^r]$ for $f \in F'$ where $\bar{z}_f^r$ is the current dimensioning and S_f^r is the maximum resource capacity of z_f^r. Proceeding normally beyond integration will yield the appropriate *extended* formulation.

D. Drawbacks of Fog-DC-MILP formulation

This formulation allows for both node and link mappings to be performed in one shot; however, since the Fog-DC-MILP formulation is based on an integer linear programming model, it suffers from scalability issues. With a large number of IoT requests, the mathematical model takes on a large number of variables and constraints. This is potentially a significant drawback to solving the MILP model optimally in a reasonable computation time.

i. For a given IoT request $I_n = (T_n, E_n)$, a virtual link $e \in E_n$ can be assigned to up to

$$|W| \times |W| \times |\Pi| \tag{32}$$

possible embedding path solutions.

ii. Thus, for $|E| = \max_{n \in N} |E_n|$ being the maximal size of virtual links, $|E|$ virtual links over N IoT requests can be assigned to up to

$$(|E| \times |W| \times |W| \times |\Pi|)^{|N|} \tag{33}$$

possible mapping solutions, which can be approximated by the exponential number $O(g^{|N|})$.

Node and link embedding is known to be an NP-hard problem, equivalent to a multi-way separator problem [46]. To address this complexity, we propose a decomposition approach based on the Column Generation technique [47]. This implies a pricing of non-basic variables to generate new columns or to prove LP optimality at a node of the branch-and-bound tree.

VI. Column Generation Formulation for an IoT service Resource Allocation (Fog-DC-CG)

To avoid the scalability issue identified in the MILP formulation, we propose to use the Column Generation formulation (Fog-DC-CG) to allocate Fog-DC resources to service IoT requests. We reformulate the resource allocation problem in terms of Independent Fog-DC Configurations (IFCs). Each IFC solves the resource allocation problem of a single IoT request. We denote by Θ the set of all possible IFCs. Accordingly, the resource allocation problem can then be formulated with respect to the decision variables λ_θ such that

$$\lambda_\theta = \begin{cases} 1, & \text{if IFC } \theta \in \Theta \text{ is used in the} \\ & \text{Fog mapping solution} \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

In this new formulation, the mapping problem is to choose a maximum of $|N|$ IFCs, as each IFC is serving one IoT request. The resulting configuration corresponds to what is known as the master problem in a column generation approach, while each configuration IFC corresponds to what is known as the pricing problem. Here we are making the assumption that parameter $A = 1$, i.e., the Fog-DC design should accept all IoT requests $n \in N$.

The IFC configuration $\theta \in \Theta$ is defined by the vector $(a_n^\theta)_{n \in N}$ such that

$$a_\theta^n = \begin{cases} 1, & \text{if IFC } \theta \text{ services the IoT request } I_n \\ 0, & \text{otherwise} \end{cases} \tag{35}$$

$$\sum_{n \in N} a_\theta^n = 1, \quad \theta \in \Theta. \tag{36}$$

We denote by COST_θ the cost of configuration θ, which corresponds to the sum of costs of the resources used (hosting and networking) for the IoT request granted by IFC θ.

The use of Column Generation formulation divides the original problem into a master problem and a pricing problem with two separate objectives:

1) Master Problem: the problem of finding the best subset among the already generated IFCs that minimize the dimensioning costs.
2) Pricing Problem: the problem of generating an additional column (IFC) to the constraint matrix of the Master Problem.

A. Master Problem

The master problem, denoted by **Fog-CG-M**, is defined as follows:

1) Objective function:

$$\min \sum_{\theta \in \Theta} \text{COST}_\theta \, \lambda_\theta + \sum_{f \in F} \sum_{r \in R} D_f^r \cdot y_f^r + u_f^r \cdot z_f^r \tag{37}$$

where

$$\text{COST}_\theta = \sum_{l \in L} B_\theta^{\text{B}}(l) \cdot c_l + \sum_{w \in W} \sum_{r \in R} P_\theta^r(w) \cdot c_w^r. \tag{38}$$

Here, c_w^r and c_l are the same unit resource costs as in Section V-B. $B_\theta^{\text{B}}(l)$ is the bandwidth used on a networking link l by IFC θ and B_l is the maximum available bandwidth on networking link l. We also denote $P_\theta^r(w)$ to be the amount of resource r in Fog-DC location w used by IFC θ.

2) IoT resource allocation constraints:

$$\sum_{\theta \in \Theta} \lambda_\theta \cdot P_\theta^r(w) \leq z_w^r; \quad w \in W, \ r \in R \qquad (\alpha_w^r) \tag{39}$$

$$\sum_{\theta \in \Theta} \lambda_\theta \cdot B_\theta^{\text{B}}(l) \leq B_l; \qquad l \in L \qquad (\beta_l) \tag{40}$$

$$\sum_{\theta \in \Theta} \lambda_\theta \Delta_\theta^{\text{L},\rho,\omega}(t) \leq \tau_t; \quad t \in T_n, n \in N \qquad (\gamma_t) \tag{41}$$

$$\sum_{\theta \in \Theta} \lambda_\theta \leq |N| \qquad (\mu_0) \tag{42}$$

$$\sum_{\theta \in \Theta} \lambda_\theta \cdot a_\theta^n \geq 1; \qquad n \in N. \qquad (\psi_n) \tag{43}$$

Equation (39) expresses the available capacity of resource r in Fog-DC node w. Equation (40) expresses the bandwidth capacity of networking link l. Equation (41) expresses the latency threshold that must be satisfied for each task t for some transmission delay, ρ-percentile of IoT traffic and ω-percentile of congestion. Equation (42) guarantees the convexity of the ILP model. Equation (43) grants the satisfaction of the maximum number of IoT requests.

3) Fog-DC Design and Dimensioning Constraints:

$$z_f^r \leq S_f^r \cdot y_f^r \, ; f \in F, r \in R \tag{44}$$

$$\sum_{f \in F} y_f^r \leq Y_{\text{MAX}}^r \, ; r \in R. \tag{45}$$

Equations (44) and (45) are the same constraints as in the Fog-DC-MILP formulation.

4) Linear Relaxation of Fog-CG-M: In order to obtain the dual variables associated with Equations (39) - (43), we formulate a linear relaxation of Fog-CG-M. This Linear Program formulation, denoted **Fog-CG-M-LP** only differs from Fog-CG-M in the removal of variables z and y. For $c \in C$, let S_c^r be the total amount of resource r in c.

$$\min \ \text{Fog-CG-M-LP} = \sum_{\theta \in \Theta} \text{COST}_\theta \lambda_\theta$$

Subject to

$$\sum_{\theta \in \Theta} \lambda_\theta \cdot P_\theta^r(w) \leq S_w^r; \quad w \in W, r \in R \tag{46}$$

$$(40) - (43)$$

$$\lambda_\theta \in [0, 1]$$

B. Pricing problem

As mentioned previously, the pricing problem corresponds to the generation of an additional configuration (IFC), i.e., an additional column for the constraint matrix of the current master problem. Let α_w^r, β_l, γ_t, μ_0, and ψ_n be the dual variables associated with constraints (39), (40), (41), (42) and (43) respectively and obtained from solving the dual of Fog-CG-M-LP. Then, the reduced cost of variable λ_θ for an IFC θ can be written:

$$\overline{\text{COST}}_\theta = \text{COST}_\theta - \sum_{n \in N} \psi_n \cdot a_\theta^n + \sum_{r \in R} \sum_{w \in W} \alpha_w^r \cdot P_\theta^r(w)$$
$$+ \sum_{l \in L} \beta_l \cdot B_\theta^{\text{B}}(l) + \sum_{t \in T_n} \sum_{n \in N} \gamma_t \cdot \Delta_\theta^{\text{L},\rho,\omega}(t) + \mu_0 \tag{47}$$

where COST_θ is defined by (38).

We now express (47) in terms of the decision variables of the pricing problem; in order to alleviate notation, we omit θ from the index of the decision variables below. Those variables are implicitly defined within the context of θ as follows.

$$a_n = \begin{cases} 1, & \text{if an IoT request } I_n \ n \in N \text{ is serviced} \\ 0, & \text{otherwise} \end{cases} \tag{48}$$

$$x_w^t = \begin{cases} 1, & \text{if virtual node } t \in T_n, \, n \in N \text{ is assigned} \\ & \text{to Fog-DC node } w; \, M_{\text{W}}(t) = w \\ 0, & \text{otherwise} \end{cases} \tag{49}$$

$$x_\pi^e = \begin{cases} 1, & \text{if virtual link } e \in E_n, \, n \in N \text{ is assigned} \\ & \text{to physical path } \pi; \, M_{\text{E}}(e) = \pi \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

Next, we derive the relations between the pricing variables and the coefficients of the master problem for each configuration $\theta \in \Theta$. For each $n \in N$, $a_\theta^n = a_n$. For each Fog-DC node $w \in W$ and resource $r \in R$, we have:

$$P_\theta^r(w) = \sum_{n \in N} \sum_{t \in T_n} p_t^r \cdot x_w^t. \tag{51}$$

For each link $l \in L$, we have:

$$B^{\mathrm{B}}_{\theta}(l) = \sum_{e=(t,t')\in E_n} \sum_{n\in N} \sum_{(w,w')\in W^2} \sum_{\pi\in\Pi^l_{(w,w')}} b_t \cdot x^{\pi}_e. \quad (52)$$

For each task $t \in T_n,\ n \in N$, we have:

$$\Delta^{\mathrm{L},\rho,\omega}_{\theta}(t) = \sum_{t\in T_n} \sum_{n\in N} \sum_{(w_1,w)\in W^2} \left(\nu^{\rho,\omega}_{t_1,w_1} \cdot x^{t_1}_{w_1} + \nu^{\rho,\omega}_{t,w} \cdot x^{t}_{w}\right) + \sum_{t\in T_n} \sum_{n\in N} \sum_{(w_1,w)\in W^2} x^{t_1}_{w_1} x^{t}_{w} \cdot \tilde{\phi}^{\rho,\omega}_{(w_1,w)} + \sum_{t\in T_n} \sum_{n\in N} \sum_{i=t_1}^{t} \zeta_i \quad (53)$$

where $\nu^{\rho,\omega}, \tilde{\phi}^{\rho,\omega}$ and ζ are respectively the transmission, routing and processing costs from (13), and t_1 represents the first task of a request.

The substitution of the Pricing decision variables into the reduced cost in (47) gives us the Pricing problem denoted as **Fog-CG-P**:

$$\begin{aligned}\min & \sum_{e=(t,t')\in E_n} \sum_{n\in N} \sum_{(w,w')\in W^2} \sum_{\pi\in\Pi^l_{(w,w')}} b_t \cdot x^{\pi}_e \cdot (c_l + \beta_l) \\ & + \sum_{n\in N} \sum_{t\in T_n} \sum_{w\in W} p^r_t \cdot x^t_w \cdot (c^r_w + \alpha^r_w) \\ & + \sum_{n\in N} \sum_{t\in T_n} \sum_{(w_1,w)\in W^2} \Delta^{\mathrm{L},\rho,\omega}_{\theta}(t) \cdot \gamma_t \\ & + \sum_{t\in T_n} \sum_{n\in N} \sum_{i=t_0}^{t} \zeta_i + \mu_0 - \sum_{n\in N} \psi_n \cdot a_n \end{aligned} \quad (54)$$

where $\Delta^{\mathrm{L},\rho,\omega}_{\theta}(t)$ is defined by (53).

The optimal solution of Fog-CG-P defines the additional configuration to be added via (48), (49) and (50).

C. Pricing Constraints

1) Mapping of IoT service tasks:

i. Mapping is done for all tasks of an accepted IoT request I_n.

$$a_n \leq \sum_{(w,w')\in W^2} x^t_w\, x^{t'}_{w'}\ ;\ e = (t,t') \in E_n, n \in N. \quad (55)$$

ii. A task t of an accepted request I_n is assigned to only one Fog-DC location node w.

$$\sum_{w\in W} x^t_w \leq a_n\ ;\ t \in T_n,\ n \in N. \quad (56)$$

2) Mapping of IoT request Link:

$$\sum_{(w,w')\in W^2} \sum_{\pi\in\Pi^e_{(w,w')}} x^e_{\pi} \leq a_n\ ; e \in E_n,\ n \in N. \quad (57)$$

$$x^t_w x^{t'}_{w'} \leq \sum_{\pi\in\Pi_{(w,w')}} x^e_{\pi}\ ; (w,w') \in W^2, \quad e = (t,t') \in E_n,\ n \in N. \quad (58)$$

Equations (57) expresses that if request I_n is accepted then at least one networking path π is assigned to grant data transfer over virtual link e, and likewise (58) for a path assignment to Fog-DC site locations w and w'.

3) Latency relaxation: To push the Column Generation formulation towards generating viable columns, we add a relaxed latency constraint. We define a variable $\tilde{\Delta}^{\mathrm{L},\rho,\omega}_{w}(t)$ such that

$$\tilde{\Delta}^{\mathrm{L},\rho,\omega}_{w}(t) = \nu^{\rho,\omega}_{t,w} x^t_w + \sum_{i=t_1}^{t} \zeta_i. \quad (59)$$

From (53), we infer that $\tilde{\Delta}^{\mathrm{L},\rho,\omega}_{w}(t) \leq \Delta^{\mathrm{L},\rho,\omega}_{w}(t)$; therefore, we add the constraint

$$\sum_{w\in W} \tilde{\Delta}^{\mathrm{L},\rho,\omega}_{w}(t) \leq \tau_t; \qquad t \in T_n,\ n \in N. \quad (60)$$

4) Linearization of Quadratic terms: We note that objective term (53) and constraints (29), (55) and (58) include the quadratic terms $x^t_w x^{t'}_{w'}$. Since this quadratic term is the product of two binary variables, it can be linearized easily by replacing quadratic term by a new binary variable $y^{t,t'}_{w,w'}$ where $y^{t,t'}_{w,w'} = x^t_w x^{t'}_{w'}$ and by adding the constraints

$$\begin{aligned} y^{t,t'}_{w,w'} &\geq x^t_w \\ y^{t,t'}_{w,w'} &\geq x^{t'}_{w'} \end{aligned} \quad (61)$$

which ensure that $y^{t,t'}_{w,w'}$ will be zero if either x^t_w or $x^{t'}_{w'}$ are zero. Adding the inequality

$$y^{t,t'}_{w,w'} \geq x^{t'}_{w'} + x^t_w - 1 \quad (62)$$

makes sure that $y^{t,t'}_{w,w'}$ will take value 1 if both binary variables x^t_w or $x^{t'}_{w'}$ are set to 1. We note that such a linearization technique is done implicitly in our simulation by the used linear solver CPLEX.

D. Solving the Fog-DC-CG Model:

The steps involved in solving the Fog-DC-CG model formulated in Section V are as follows:

1) Initialize Fog-CG-M-LP by a subset of dummy configuration that is, a set of artificial IFCs with a large cost.
2) Solve the dual of Fog-CG-M-LP formulation to optimality using CPLEX solver to obtain dual variables $\alpha^r_w, \beta_l, \gamma_t, \psi_n$ and μ_0, the variables associated with the Fog-CG-M-LP constraints.
3) Solve the Pricing problem Fog-CG-P to optimality using CPLEX solver. This may generate several possible columns (IFCs).
4) For each column generated, calculate the Reduced Cost. If a column with a negative reduced cost has been found, add this column to the current master problem and repeat Steps 2 and 3. Otherwise, Fog-CG-M-LP is optimally solved.

The optimal solution of Fog-CG-M-LP only provides a lower bound on the optimal integer solution of Fog-CG-M. We solve the Fog-CG-M integer programming formulation to optimality using Branch and Bound CPLEX solver.

VII. Benchmarks

To make an appropriate comparison between our two proposed models, we define two benchmarks inspired by literature: a matching-based model [26] and a greedy model [48] known as Fog-DC-Match and Fog-DC-Greedy respectively.

A. *Fog-DC-Match*

For a request $t \in T_n, n \in N$, let $r^t_{\mathrm{W}} : \mathbb{Z}^+ \mapsto W$ be a function such that $r^t_{\mathrm{W}}(i)$ returns the Fog-DC node with ith lowest latency to t. We formulate a relaxed MILP to find the Fog-DC nodes that minimize the ranking as the first phase of our heuristic.

$$\tilde{Z}_{P1} = \min \sum_{n\in N}\sum_{t\in T_n}\sum_{i=1}^{|W|} i \cdot x^t_{w_i}; \quad w_i = r^t_{\mathrm{W}}(i) \tag{63}$$

Subject to

$$\sum_{n\in N}\sum_{t\in T_n} x^t_{w_i}\cdot p^r_t \leq S^r_{w_i}; \tag{64}$$

$$r \in R,\, w_i = r^t_{\mathrm{W}}(i) \in W$$

$$x^t_f \cdot p^r_t \leq S^r_f \cdot y^r_f; \tag{65}$$

$$f = r^t_{\mathrm{W}}(i) \in F, i \in \{1, ..., |W|\},\ r \in R$$

$$(21) - (23).$$

For a virtual link $e \in E_n, n \in N$, let $r^e_{\mathrm{E}} : \mathbb{Z}^+ \mapsto \Pi$ be a function such that $r^e_{\mathrm{E}}(i)$ returns the Fog-DC path with ith lowest cost, given the set optimal Fog-DC node mappings from the first phase denoted $M_{\mathrm{W,P1}} : T_n \mapsto W$, we rank the paths to formulate our second phase of our heuristic.

$$\tilde{Z}_{P2} = \min \sum_{n\in N}\sum_{e\in E_n}\sum_{i=1}^{|\Pi|} i \cdot x^e_{\pi_i}; \quad \pi_i = r^e_{\mathrm{E}}(i) \tag{66}$$

Subject to

$$(24), (27)$$

$$x^t_w \cdot x^{t'}_{w'} = \sum_{\pi\in\Pi_{(w,w')}} x^e_\pi; \tag{67}$$

$$e = (t, t') \in E_n, n \in N,$$

$$M_{\mathrm{W,P1}}(t) = w,\ M_{\mathrm{W,P1}}(t') = w'.$$

For design and dimensioning solutions from $\tilde{Z}_{P1}$ and $\tilde{Z}_{P2}$, we know $z^r_f = \sum_{t\in T_n}\sum_{n\in N} x^t_f \cdot p^r_t,\ f \in F$ so we can obtain the objective function Z_{MILP} (20) for comparison with other models.

B. *Fog-DC-Greedy*

For a set of ordered tasks $\{t_1, ..., t_{q_k}\}$ from a request $n \in N$, we wish to allocate task resources to Fog-DC nodes by greedily choosing the node w_j with available resources and with minimal latency to t_j . Once a servicing Fog-DC node is chosen, we find a path with the minimal routing cost between w_{j-1} and w_j that satisfies the bandwidth constraints per link. Aside from reducing design and dimensioning cost, our main objective is to satisfy latency requirements of tasks for a high Fog acceptance rate. For this reason, our greedy metric is first and foremost on latency.

Algorithm 2 Fog-DC-Greedy; Greedy algorithm by Fog-DC node latency.

```
Result: Greedy Fog-DC Design and Dimensioning.
Enumerate requests such that $N = \{1, 2, ..., |N|\}$.
 For request $n$, enumerate tasks $t_1, ..., t_{q_k}$ where $o_{t_i} < o_{t_j}$,
 $1 \leq i < j \leq q_k$, $M_{\mathrm{K}}(n) = k$.
 Initialize $\omega \in (0, 1)$, $S^r_{w,\mathrm{USED}} = 0, B_{l,\mathrm{USED}} = 0$.
 for $n$ = 1 to $|N|$ do
     $k \leftarrow M_{\mathrm{K}}(n)$
     for $j = 1$ to $q_k$ do
         $W' \leftarrow W$;
         // For task $t_j$, select a node $w_j$ with minimal latency.
         while True do
             if $j = 1$ then
                 $\Delta_{t_1} \leftarrow \min\{\nu^{\rho,\omega}_{t_1,w} \mid w \in W'\}$;
                 $w_1 \leftarrow \arg\min\{\nu^{\rho,\omega}_{t_1,w} \mid w \in W'\}$;
             end
             else
                 $\Delta_{t_j} \leftarrow \min\{\Delta^{t_j,\rho,\omega}_{(w_1,w)} \mid w \in W'\}$;
                 $w_j \leftarrow \arg\min\{\Delta^{t_j,\rho,\omega}_{(w_1,w)} \mid w \in W'\}$;
             end
             if $p^r_{t_j} \leq S^r_{w_j} - S^r_{w_j,\mathrm{USED}}\ \forall r \in R$ then
                 $S^r_{w_j,\mathrm{USED}} \leftarrow p^r_t + S^r_{w_j,\mathrm{USED}}$;
                 // Greedily route backwards to $w_{j-1}$.
                 while $j > 1$ do
                     $c_\pi \leftarrow \min_{\pi\in\Pi_{(w_{j-1},w_j)}} \sum_{l\in\pi} c_l b_l$;
                     $\pi \leftarrow \arg\min_{\pi\in\Pi_{(w_{j-1},w_j)}} \sum_{l\in\pi} c_l b_l$;
                     if $b_l \leq (B_l - B_{l,\mathrm{USED}}))$, $\forall l \in \pi$ then
                         $B_{l,\mathrm{USED}} \leftarrow b_l + B_{l,\mathrm{USED}}, \forall l \in \pi$;
                         break;
                     end
                     else
                         $\Pi_{(w_{j-1},w_j)} \leftarrow \Pi_{(w_{j-1},w_j)} \setminus \pi$;
                     end
                 end
             end
             else
                 $W' \leftarrow W' \setminus w_j$;
             end
         end
     end
 end
```

VIII. Results

A. Simulation Setup

In this section we conduct a time and cost comparison between the Fog-DC-MILP model, and the heuristic Fog-DC-CG model. Both models are solved using IBM CPLEX Solver on a machine with an i7 Dual Core 2.5GHz CPU and 12GB of RAM. In addition, we compare the results with the two heuristics Fog-DC-Match and Fog-DC-Greedy using the same configurations.

Given the intractability of Fog-DC-MILP, we selected the following configuration settings to produce results in a reasonable amount of time. We chose to use 5 IoT requests with varying number of tasks from 1 to 4, totaling 10 tasks in all. For simplicity, we set the acceptance ratio (21) threshold to be 1. For scalability, we added an arrival rate $\lambda \in [1, 50]$ to inflate the number of requests, and consequently the number of tasks. As noted in (3), an increase in the IoT traffic volume percentile ρ leads to an increase in the arrival rate λ by similar factors; therefore, we simplify the simulation by only including a varying arrival rate λ. The service rate μ_w for a Fog-DC node $w \in W$ was selected in the range $\mu_w \in [10, 50]$ milliseconds. We set the congestion percentile ω to 0.5.

Over the set of tasks, resource requirements were mostly selected from a uniform distribution in $[a^r_{\mathrm{T}}, b^r_{\mathrm{T}}]$ with low and high values of $a^{\mathrm{CPU}}_{\mathrm{T}} = 0$ GHz and $b^{\mathrm{CPU}}_{\mathrm{T}} = 2.5$ GHz for CPU, $a^{\mathrm{MEM}}_{\mathrm{T}} = 0$ GB and $b^{\mathrm{MEM}}_{\mathrm{T}} = 2.5$ GB for MEM, and $a^{\mathrm{STR}}_{\mathrm{T}} = 0$ GB and $b^{\mathrm{STR}}_{\mathrm{T}} = 15$ GB for STR; for each resource, a high resource requirement was assigned to a task above 10 GHz, 10 GB and 100 GB for CPU, MEM, and STR respectively. We set the latency requirement for each task at 500 to 1100 milliseconds to simulate IoT time-sensitivity. We set $s^+_t \in (0, 10]$ and $s^-_t \in (0, 10]$ in megabytes, bandwidth requirement $b_t \in (0, 10]$ megabytes, and the processing time $\xi_t \in [10, 100]$ milliseconds.

We chose to design a Fog infrastructure with 29 candidate Fog locations. Similar to the IoT settings, the resource capacities for Fog nodes were chosen from a uniform distribution range $[a^r_{\mathrm{F}}, b^r_{\mathrm{F}}]$ with low and high values of $a^{\mathrm{CPU}}_{\mathrm{F}} = 0$ and $b^{\mathrm{CPU}}_{\mathrm{F}} = 25$ for CPU, $a^{\mathrm{MEM}}_{\mathrm{F}} = 0$ and $b^{\mathrm{MEM}}_{\mathrm{F}} = 125$ for MEM, and $a^{\mathrm{STR}}_{\mathrm{F}} = 0$ and $b^{\mathrm{STR}}_{\mathrm{F}} = 1000$ for STR. For each resource, at least one Fog location has a resource capacity of zero; we acknowledge that a Fog node may not have the capability for every resource. The network architecture of candidate Fog nodes are shown in Fig. 6.

We designed the infrastructure to include two fixed Cloud Data Centres, with capacity in $[a^r_{\mathrm{C}}, b^r_{\mathrm{C}}]$ with $b^r_{\mathrm{F}} \ll b^r_{\mathrm{C}}$ to allow Cloud nodes to accept any task not accepted by Fog nodes. The low and high values selected were $a^{\mathrm{CPU}}_{\mathrm{C}} = 100000$ and $b^{\mathrm{CPU}}_{\mathrm{C}} = 200000$ for CPU, $a^{\mathrm{MEM}}_{\mathrm{C}} = 51200$ and $b^{\mathrm{MEM}}_{\mathrm{C}} = 102800$ for MEM, and $a^{\mathrm{STR}}_{\mathrm{C}} = 500000$ and $b^{\mathrm{STR}}_{\mathrm{C}} = 1000000$ for STR.

As noted in Fig. 3, a Fog-DC node may be accessed directly by a task t if the distance is no more than γ_t, or via an access point that routes through other internet channels. Equation (7) uses an upper bound $h_{t,w}$ on the number of hops between task t and Fog-DC node w, however these may be through access points and not the Fog-DC infrastructure.

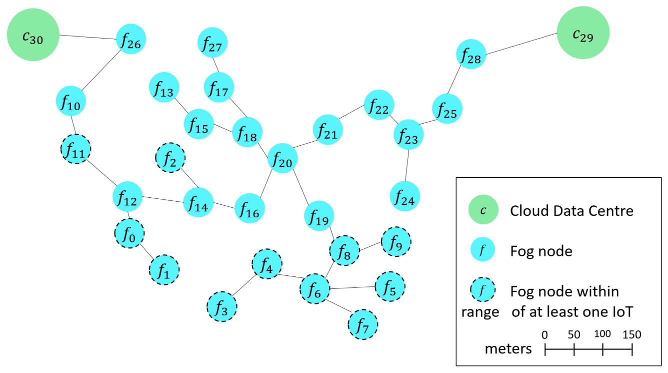


Fig. 6: Network architecture of candidate Fog nodes used for comparison.

For two IP addresses, the number of hops between them for a single ping can be determined using the `traceroute` command. We tested several IPs at different distances from ourselves to estimate $h_{t,w}$ based on distance $d_{t,w}$. We found for $d_{t,w} \in (0, 10]$, $h_{t,w} \leq 5$ and for $d_{t,w} \in [100, 1000]$, $h_{t,w} \leq 14$; our findings are supported by [45]. Since all the Fog node candidates are within 10km of the tasks, we set $h_{t,f} = 5$ for $f \in F$. For Cloud nodes $c \in C$ we let $h_{t,c} = \max\{5, d_{t,c}/10\}$. The maximum number of hops between two Fog-DC nodes $h_{w,w'}$ is defined as the maximum number of links for a path $\pi \in \Pi_{(w,w')}$ in the Fog-DC architecture of Fig. 6.

B. Dimensioning/Partitioning Scheme Comparison

Fig. 7 shows the optimal design and dimensioning solutions obtained for Fog-DC-MILP vs. Fog-DC-CG. For each model in Fig. 7, the dimensioned nodes are identified, the resource utility is detailed, and the used paths are shown. Given that the portrayed network architecture in Fig. 7 is based on assumed latitude and longitude of Fog nodes, we can infer that Fog nodes in similar areas are dimensioned, allowing services to be provided to IoT devices in the same regions with real-time responses.

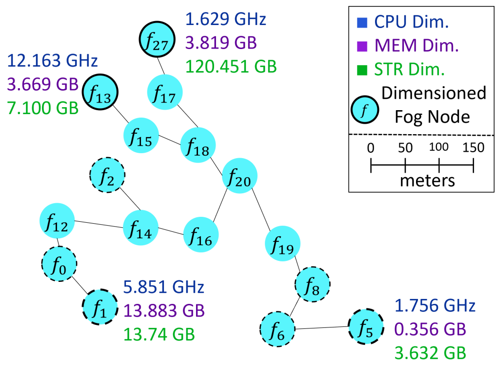
Considering the Fog nodes with differing dimensioning solutions in Table V, we observe the similarities in both solutions with the smallest difference being in MEM allocation and the largest difference in STR allocation. Based on our map scale in Fig. 6 and 7, f_5 and f_9 are approximately 25 meters apart, whereas f_1 is approximately 500 meters from both; this larger distance is still close enough to allow for the benefits of minimal latency between an IoT device that sits 500 meters from any Fog node, making the Fog-DC-CG solution of allocating more STR resources in f_9 instead of f_1 a reasonable change that does not change the latency viability of our current configuration and IoT traffic predictions; however, this change may affect IoT traffic that is higher than the ρ-percentile of data for which we have accounted, or new IoT devices on the edge of our designed area.

C. Dimensioning Cost

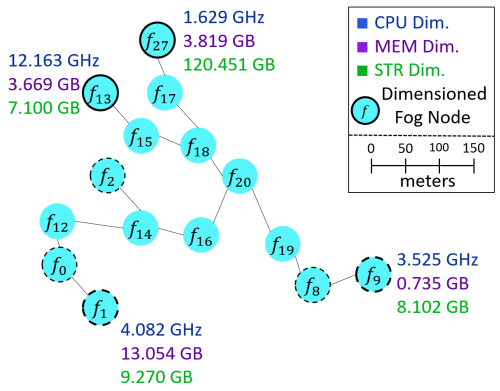
By increasing the arrival rate of each IoT request class, we can increase the total number of tasks arriving to the

TABLE V: Dimensioning solutions of Fog-DC-MILP vs Fog-DC-CG (Fig. 7) for Fog nodes in similar areas.

Fog	CPU	MEM	STR
f_1 (MILP)	5.851 GHz	13.883 GB	13.774 GB
f_1 (CG)	4.082 GHz	13.054 GB	9.270 GB
f_5 (MILP)	1.756 GHz	0.356 GB	3.632 GB
f_9 (CG)	3.525 GHz	0.735 GB	8.102 GB



(a) Fog-DC-MILP



(b) Fog-DC-CG

Fig. 7: The mapping solution for the MILP and CG formulation over 5 Requests and 10 tasks.

Fog-DC system to compare the four models given greater IoT traffic. We are most interested in whether Fog-DC-CG can approximate the optimal MILP model in cost with a substantial decrease in time. Table VI shows that Fog-DC-CG attains a near-optimal design and dimensioning cost of the Fog Infrastructure, with the cost difference below 3% for up to 80 tasks. The absolute difference varies between 5.889 and 10.480, but does not increase monotonically with increasing IoT traffic, leading us to hypothesize that the percentage difference gradually decreases with increasing number of tasks. A more expansive study into the cost differences of MILP and CG approaches is left for future work. Fig. 8 shows the significantly reduced cost of Fog-DC-CG compared to the other two heuristics Fog-DC-Match and Fog-DC-Greedy.

D. Computation Time

Our performance comparison of Fog-DC-MILP and Fog-DC-CG in Fig. 9 shows that Fog-DC-CG calculates a dimensioning solution in significantly reduced time. While Fog-DC-Match performs performs moderately better than Fog-DC-MILP, the computation time is not scalable for higher number of tasks. In Fig. 9, we executed Fog-DC-MILP for at most 80 tasks as a higher number of tasks proved computationally infeasible, whereas Fog-DC-Match became computationally infeasible after around 300 tasks. We were able to execute Fog-DC-CG for up to 500 tasks within a practical and scalable amount of time with a near-linear time growth. Though Fog-DC-Greedy is extremely fast, Fig. 8 shows that it is also the worst performing by cost.

Fig. 9b is identical to Fig. 9a with the x-axis restricted to $[0, 100]$. This allows us to see the similar time performances between Fog-DC-CG, Fog-DC-Match and Fog-DC-Greedy for low number of tasks. We also see the point at which the three models begin to deviate from each other in performance. In both cases, Fog-DC-MILP is growing at an alarming rate from the beginning.

To better observe the performance of Fog-DC-CG, we simulated 600 independent Fog-DC system configurations. Each configuration had 10 to 100 Fog candidates in different topological organizations, 10 to 50 IoT requests, and 1 to 5 tasks per request. The latitude and longitude of the Fog and IoT devices were selected uniformly in a selected region of radius of 5 kilometers, affecting the reachability of Fog nodes from IoT for each simulated instance.

The solution time per Fog-DC configuration setup are shown in Fig. 10a and 10b with either number of IoT requests or Fog candidates on the x-axis, and the other metric as a colourmap. Fig. 10a shows that for at most 50 IoT requests, the number of requests does not have a strong influence on the solution time; no evident correlation is observed, with only the Fog colourmap having a clear pattern of increasing with solution time. On the other hand, Fig. 10b shows that it is the number of Fog candidates that largely dictates the solution time. These observations are further solidified by the correlations calculated in Table VII that shows the number of Requests and of Fog candidates respectively have a correlation of 0.478 and 0.829 with solution time. Based on the results of Fig. 10c with both the number of Fog candidates and IoT requests, we observe some scalability with increasing Fog and IoT devices, though further simulation are needed. Referring to Fig. 10, no clear linearity is observed in any of the metrics; we leave further simulation and statistical analysis for future work.

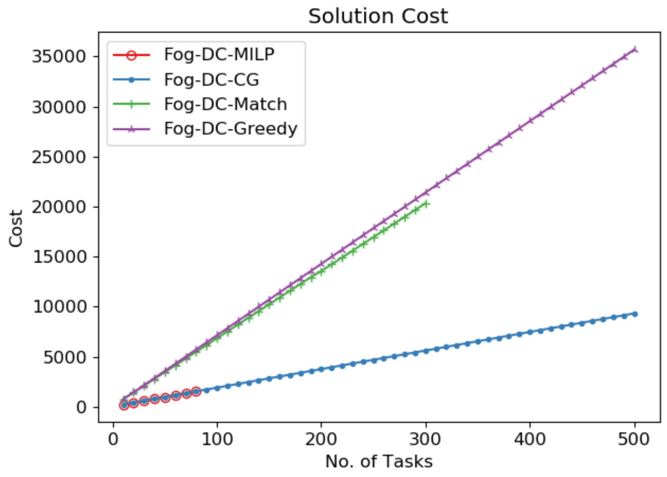
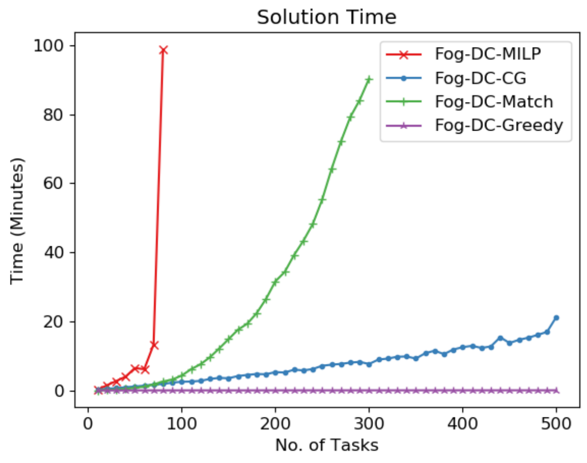


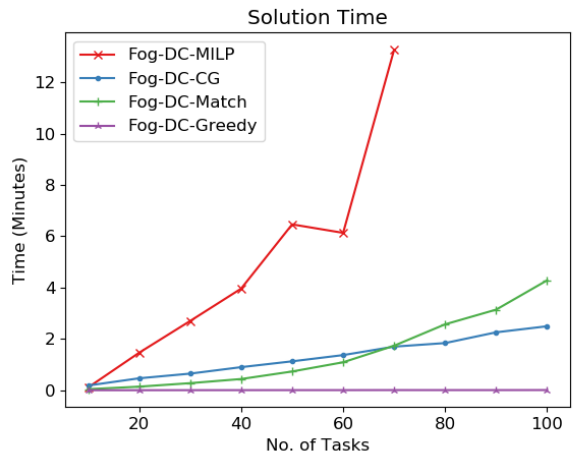
Fig. 8: Objective cost comparison of Fog-DC-MILP, Fog-DC-CG, Fog-DC-Match, and Fog-DC-Greedy.

TABLE VI: Cost comparison in percentage between Fog-DC-MILP and Fog-DC-CG.

No. of Tasks	Fog-DC-MILP Solution	Fog-DC-CG Solution	% Gap	Gap
10	214.977	220.866	2.739%	5.889
20	402.828	411.208	2.080%	8.380
30	590.348	596.716	1.078%	6.368
40	773.547	783.713	1.314%	10.166
50	958.205	968.259	1.049%	10.051
60	1144.030	1153.480	0.826%	9.150
70	1331.56	1342.040	0.787%	10.480
80	1518.440	1529.040	0.698%	10.600

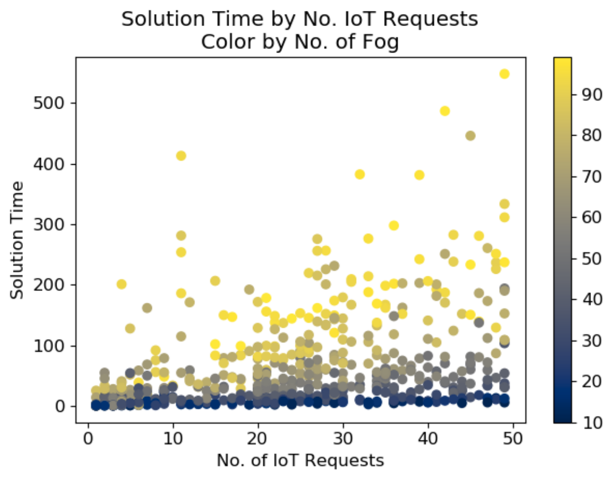


(a) Full scale

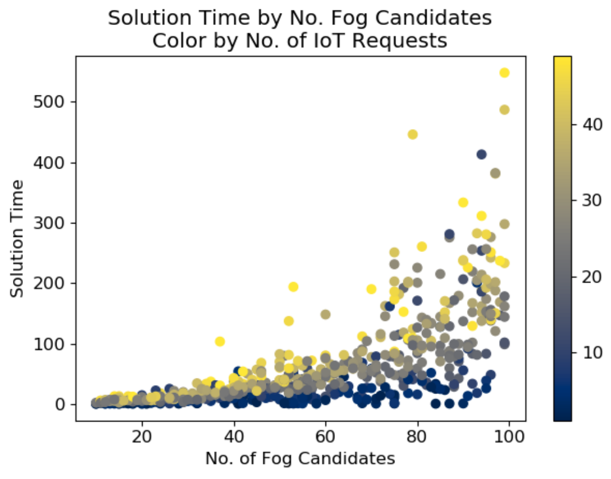


(b) Small scale

Fig. 9: Solution time comparison of Fog-DC-MILP and Fog-DC-CG.



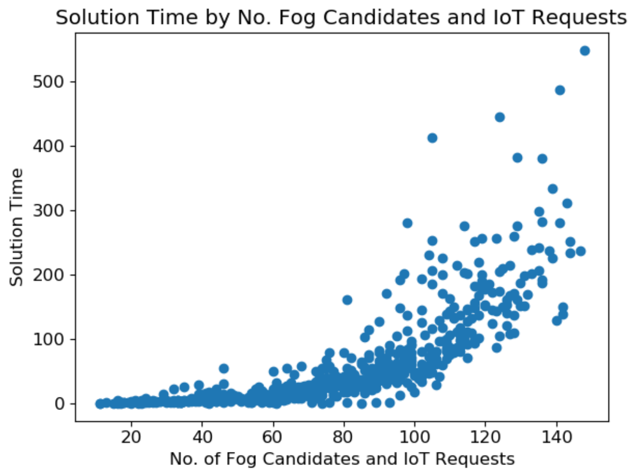
(a) Requests with Fog colourmap



(b) Fogs with Request colourmap

TABLE VII: Spearman's Correlation of simulated solutions by number of Requests and Fog Candidates used.

Correlation against Time	Spearman's Correlation
No. of Requests vs. Time	0.496
No. of Fog Candidates vs. Time	0.805



(c) Fogs and Requests

Fig. 10: Solution time of Fog-DC-CG by No. of IoT Requests and Fog Candidates.

IX. Conclusion & Future Work

In this paper, we have proposed an optimal design and dimensioning formulation of Fog infrastructure using MILP to minimize infrastructure cost. To overcome scalability issues while keeping cost effectiveness, we proposed a near-optimal Column Generation formulation. Simulation results show that design and dimensioning solution are with under 3% difference from the optimal solution with significantly reduced computation time. Results also show Column Generation cost is much lower than matching-based and greedy heuristics. Simulation and analysis of Fog-DC configurations conclude computation time is highly correlated with number of Fog candidates, and moderately correlated with IoT requirements.

In future work, we propose to determine more accurate estimation of network congestion resulting from fluctuating IoT traffic by means of a simulation toolkit such as iFogSim [39], which also leads to a more accurate estimation of transmission time for all tasks in a request. Due to intractability of MILP, we could not perform large simulations; we plan to use geographic zoning techniques to further reduce the time complexity of our MILP and CG models. We can then perform a larger scale simulation and comparison of MILP and CG models over a wider variety of current IoT and Fog technologies. Our current proposition allows for extensibility of a Fog design and dimensioning scheme assuming all else remains constant; we propose to look further at extensibility of existing/ modified Fog infrastructures to add greater coverage and/or resource availability for an increase in IoT traffic. Given our simulation results of computation time per Fog-DC configurations (see Fig. 10), our relationships are not linear; we intend to do further statistical studies with the goal of predicting the expected solution time given a set of Fog-DC configurations. This would solidify our approach as a means for real and practical design, dimensioning, and future deployment.

References

[1] CISCO, "Fog computing and the internet of things: Extend the cloud to where the things are," tech. rep., 2015. Accessed: 2019-04.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[3] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for iot services in the fog," in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*, pp. 32–39, IEEE, 2016.

[4] M. Asif-Ur-Rahman, F. Afsana, M. Mahmud, M. S. Kaiser, M. R. Ahmed, O. Kaiwartya, and A. James-Taylor, "Towards a heterogeneous mist, fog, and cloud based framework for the internet of healthcare things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4049–4062, 2018.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[6] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 112–121, 2014.

[7] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2017.

[8] B. Li, Z. Fei, and Y. Zhang, "Uav communications for 5g and beyond: Recent advances and future trends," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2241–2263, 2018.

[9] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE internet of things journal*, vol. 3, no. 2, pp. 161–169, 2015.

[10] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A platform as-a-service for hybrid cloud/fog environments," in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 1–7, IEEE, 2016.

[11] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.

[12] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, pp. 258–269, ACM, 2016.

[13] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2015.

[14] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn, *et al.*, "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures," *Journal of Cloud Computing*, vol. 7, no. 1, p. 16, 2018.

[15] S. S. Gill, R. C. Arya, G. S. Wander, and R. Buyya, "Fog-based smart healthcare as a big data and cloud service for heart patients using iot," in *International Conference on Intelligent Data Communication Technologies and Internet of Things*, pp. 1376–1383, Springer, 2018.

[16] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "ifogstor: an iot data placement strategy for fog infrastructure," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pp. 97–104, IEEE, 2017.

[17] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor, "Handling service allocation in combined fog-cloud scenarios," in *2016 IEEE international conference on communications (ICC)*, pp. 1–5, IEEE, 2016.

[18] V. Cardellini, V. Grassi, F. L. Presti, and M. Nardelli, "On qos-aware scheduling of data stream applications over fog computing infrastructures," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, pp. 271–276, IEEE, 2015.

[19] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pp. 105–110, IEEE, 2015.

[20] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized iot service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017.

[21] K. Intharawijitr, K. Iida, and H. Koga, "Analysis of fog model considering computing and communication latency in 5g cellular networks," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 1–4, IEEE, 2016.

[22] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[23] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149–160, 2019.

[24] M. Sookhak, F. R. Yu, Y. He, H. Talebian, N. S. Safa, N. Zhao, M. K. Khan, and N. Kumar, "Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 55–64, 2017.

[25] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[26] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.

[27] C. Chang, S. N. Srirama, and R. Buyya, "Indie fog: An efficient fog-computing infrastructure for the internet of things," *Computer*, vol. 50, no. 9, pp. 92–98, 2017.

[28] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[29] M. Taneja and A. Davy, "Resource aware placement of iot application modules in fog-cloud computing paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 1222–1228, IEEE, 2017.

[30] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, and F. Desprez, "Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed iot applications in the fog," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pp. 751–760, ACM, 2018.

[31] M. Aazam and E.-N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for iot," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pp. 687–694, IEEE, 2015.

[32] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "Qos-aware dynamic fog service provisioning," *arXiv preprint arXiv:1802.00800*, 2018.

[33] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, 2019.

[34] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, "Foggy: a framework for continuous automated iot application deployment in fog computing," in *2017 IEEE International Conference on AI & Mobile Services (AIMS)*, pp. 38–45, IEEE, 2017.

[35] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos, "Fog based framework for iot service provisioning," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, IEEE, 2019.

[36] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," *ieee internet of things journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[37] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. Leung, "Masm: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Transactions on Industrial Informatics*, 2019.

[38] W. Zhang, Z. Zhang, S. Zeadally, and H.-C. Chao, "Efficient task scheduling with stochastic delay cost in mobile edge computing," *IEEE Communications Letters*, vol. 23, no. 1, pp. 4–7, 2018.

[39] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "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*, vol. 47, no. 9, pp. 1275–1296, 2017.

[40] F. A. Salaht, F. Desprez, A. Lebre, C. Prud'Homme, and M. Abderrahim, "Service placement in fog computing using constraint programming," in *IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING*, 2019.

[41] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[42] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, "Fundamentals of queueing theory," ch. 1.4, 3.2, pp. 10,77, John Wiley & Sons, 5th ed., 2018.

[43] J. S. W. Robert S. Witte, *Statistics*, ch. 9.6. John Wiley & Sons, 11th ed., 2016.

[44] B. Latré, P. De Mil, I. Moerman, N. Van Dierdonck, B. Dhoedt, and P. Demeester, "Maximum throughput and minimum delay in ieee 802.15. 4," in *International Conference on Mobile Ad-Hoc and Sensor Networks*, pp. 866–876, Springer, 2005.

[45] Y. Wang, P. Lai, and D. Sui, "Mapping the internet using gis: The death of distance hypothesis revisited," *Journal of Geographical Systems*, vol. 5, no. 4, pp. 381–405, 2003.

[46] D. G. Andersen, "Theoretical approaches to node assignment," 2014.

[47] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[48] A. Federgruen and H. Groenevelt, "The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality," *Operations research*, vol. 34, no. 6, pp. 909–918, 1986.

Chapter 4

Robust and Fault-Tolerant Fog Design & Dimensioning for Reliable Operation

Abstract Internet of Things (IoT) applications depend on reliable external storage and processing such as Cloud data centres. In response to high latency from Cloud, fog-computing has been introduced as a network of microdata centres closer to IoT devices that provides a geo-distributed low-latency response. Current contributions regarding design & dimensioning of fog infrastructures are developed to service a static set of IoT traffic and a reliable fog network. However, these designs are not fault-tolerant. This article explores the implementation of reliable and fault-tolerant fog infrastructures via dynamically available fog nodes – standby nodes which activate when a nearby fog node fails. We formulate the design & dimensioning of dynamically available nodes as a set partitioning problem, which is solved via a mixed-integer linear program (MILP). This MILP formulation proves to be intractable; we therefore introduce a column generation approach to increase scalability with little loss to optimal design & dimensioning cost. Compared to other benchmark heuristic methods, our column generation approach yields reduced cost, with proportional solution time.

This chapter extends the previous chapter by considering fog node failures to design & dimension a fault-tolerant, reliable fog infrastructure. This chapter has been published in IEEE Internet of Things [40].

Robust and Fault-tolerant Fog Design & Dimensioning for Reliable Operation

Ismael Martinez, Abdelhakim Senhaji Hafid, Michel Gendreau

***Abstract*—Internet of Things (IoT) applications depend on reliable external storage and processing such as Cloud data centres. In response to high latency from Cloud, fog-computing has been introduced as a network of microdata centres closer to IoT devices that provides a geo-distributed low-latency response. Current contributions regarding design & dimensioning of fog infrastructures are developed to service a static set of IoT traffic and a reliable fog network. However, these designs are not fault-tolerant. This article explores the implementation of reliable and fault-tolerant fog infrastructures via dynamically available fog nodes – standby nodes which activate when a nearby fog node fails. We formulate the design & dimensioning of dynamically available nodes as a set partitioning problem, which is solved via a mixed-integer linear program (MILP). This MILP formulation proves to be intractable; we therefore introduce a column generation approach to increase scalability with little loss to optimal design & dimensioning cost. Compared to other benchmark heuristic methods, our column generation approach yields reduced cost, with proportional solution time.**

***Index terms*—** Fault-tolerance, fog-computing, column generation, design & dimensioning, Internet of Things (IoT), mixed-integer linear programming (MILP).

I. Introduction

The number of Internet of Things (IoT) applications has been increasing drastically each year, with current estimates by International Data Corporation (IDC) to exceed 41 billion IoT devices, and generate over 79 zettabytes of data by the year 2025 [1]. Though Cloud-computing has provided means for reliable and large-scale IoT processing and data storage, Cisco emphasizes that Cloud data centres are not equipped for this large increase in IoT traffic [2].

The fog-computing paradigm is a collection of highly virtualized and geo-distributed resources on the network edge [3]. In contrast to Cloud data centres which may provide high latency to IoT applications [2], fog provides prompt service for latency-sensitive IoT requests. These include applications in health care [4], autonomous vehicles [5], and multimedia [6].

Different aspects of end-to-end fog implementation have been studied; however, there is a lack of cohesion among fog design, resource management and infrastructure evaluation studies [7]. Fog nodes may fail, which would require additional fog nodes to host/execute all IoT processes previously allocated to the failed nodes. We extend our previous work on fog design & dimensioning [8] to create a design that is robust against multiple fog node failures. We use dynamically available fog nodes, or simply *dynamic* fog nodes, as on-demand nodes that activate when a failure is detected.

I. Martinez is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, H3C 3J7, Canada (e-mail: ismael.martinez@umontreal.ca).

A. S. Hafid is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, H3C 3J7, Canada (e-mail: ahafid@iro.umontreal.ca).

M. Gendreau is with the Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Quebec, H3C 3A7, Canada (email:michel.gendreau@polymtl.ca)

The reliability of a fog system considers the reliability of (a) fog hardware/software, (b) IoT devices, applications and requests, (c) controllers for fog monitoring and resource management, and (d) the network supporting IoT-fog communication [9]. Other contributions focus on the fault detection [10], [11] and data restoration [12], [13] of IoT by fog. This paper proposes the design of a dynamic fog infrastructure that is populated by restored IoT data after a fog node failure is detected.

To the best of our knowledge, this article is the first to address the reliable and fault-tolerant design of fog infrastructure amidst uncertain network operations. Our previous approach makes a percentile estimate of IoT traffic from which a static fog design is constructed [8]. This static fog design is extended under the assumption of possible fog node failures to optimize the cost of design & dimensioning of dynamic fog nodes in the static fog infrastructure. This optimization is formulated as a set partitioning problem [14] via two approaches.

1) We propose an exact optimization approach (fog-RO-MILP) to the reliable design & dimensioning of a dynamic extension to our static fog infrastructure. This approach minimizes the implementation cost of dynamic fog nodes, while maintaining a low probability of experiencing more than one fog node failure among nodes associated with each dynamic fog node.
2) We propose a scalable heuristic column generation [15] approach (fog-RO-CG) to approximate the optimal implementation cost of the intractable fog-RO-MILP formulation. We simulate and compare results of our two methods, as well as a benchmark greedy solution, to show the long-term computational efficiency, and solution cost tradeoff of fog-RO-CG.

Current contributions of fog design make use of ideal network and IoT conditions [5], [8] that may only hold true for small fog implementations. The addition of dynamic fog nodes enhances current fog infrastructure to be robust, reliable and fault-tolerant. The scalable formulation of reliable fog design will further promote the adoption and implementation of fog networks for the benefit of latency-sensitive IoT.

The remainder of this article is organized as follows. Section II reviews related literature. Section III models the IoT-fog network components and interactions upon which we derive our optimization models. Section IV models the network latency using stochastic and queuing theory techniques. Section V introduces fog-RO-MILP for the optimization of the cost-efficient and fault-tolerant design & dimensioning of a fog infratructure. Section VI linearizes the non-linear decision variables and constraints. Section VII introduces fog-RO-CG for the scalable approximation to our exact formulation. Section VIII validates the robust infrastructure over varying network conditions, and compares design simulation results of both methods alongside a benchmark greedy solution. Section IX concludes the paper.

II. Related Work

In developing our reliable and fault-tolerant fog design approach, we consider current literature in a) fog design & dimensioning, b) dynamic network design, and c) resource offloading in dynamic fog environments. First, we identify the consequences and limitations of current fog design & dimensioning models that assume fault-tolerant fog and static IoT traffic. Second, we review robust and stochastic optimization techniques for general dynamic network design. Third, we study fog mechanisms for offloading resources to nearby on-demand fog nodes. Finally, we propose a dynamic fog design & dimensioning approach that addresses the limitations and capitalizes on the strengths of current contributions.

A. Fog Design & Dimensioning

A fog infrastructure is planned and implemented by considering a) *design* — the location of fog nodes, and b) *dimensioning* — the quantity of resources per fog node. Yu et al. [5] propose a design & dimensioning model for Internet of Vehicles (IoV) environments by constructing a network of Road-Side Units (RSUs), fog nodes, and network gateway nodes that minimizes implementation cost. The design and dimensioning of fog nodes uses a discrete set of candidate locations and resource quantities. This fog infrastructure aims to support autonomous vehicles for real-time sharing of road-traffic conditions, leading to improved traffic flow and crash avoidance. Martinez et al. [8] propose a design & dimensioning model for the general IoT landscape. The future IoT traffic and resulting network congestion is estimated stochastically, and a deterministic set of IoT requests is taken as a large percentile of the traffic distribution. An exact MILP formulation, and a heuristic column generation formulation result in similar solution costs, demonstrating the efficiency of the column generation formulation. Fog node location candidates remain discrete, whereas the resource quantities are continuous.

In [5], the estimated vehicular traffic is assumed to be static, which is not true in practice. In [8], the IoT traffic set is also static though it is a percentile of a stochastic traffic distribution, allowing for easy modification of percentile parameters to increase IoT traffic. In both cases, the design & dimensioning solution provides the smallest possible infrastructure to service the estimated IoT traffic. Overestimation of IoT traffic may lead to wasted cost from deployed but unused fog resources, while underestimation may result in a fog infrastructure incapable of servicing all IoT requests. Furthermore, the designed infrastructure is assumed to be reliable and fault-tolerant, which may not be true in practice. In neither [5] and [8] is the sensitivity of fog design & dimensioning for fluctuating IoT traffic and network congestion explored.

B. Dynamic Network Design

When formulating network design problems with stochastic elements, it is convenient to convert the stochasticity into deterministic elements. For dynamic demand of the network with known or approximated probability distribution, chance constraints enforce that demand is supported above a prescribed level of probability [16]. Supporting worst-case demand via a robust optimization approach produces a solution that is viable for all levels of demand, however may induce high costs for a largely unused network [17], [18]. Introducing dynamically available nodes to complement a fixed network can increase network supply during periods of high demand, and conserve energy and network cost during periods of low demand [19].

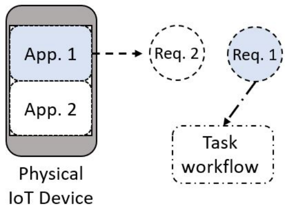
C. Resource offloading in dynamic fog environments

The concept of using dynamically available fog nodes to support an overloaded network has already been proposed and validated [20], [21]. An efficient on-demand fog node activation/creation strategy allows for the dynamic scaling of the fog network to address different levels of IoT traffic [21]. Fog infrastructures may have dedicated dormant fog nodes for on-demand support, or leverage active and nearby fog-capable devices, such as smart cars [20].

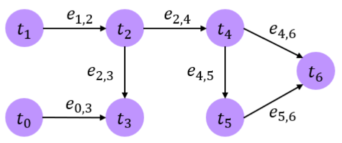
A dynamic fog node is activated when the fog is overloaded, such as when a static fog node fails. Fog systems are composed of fog nodes managed by different service providers [3]. Therefore, it is not possible to detect a failure from one central entity. Statistical methods can be used to monitor changes in system operation, and detect possible component failures [22]. Currently, fog-based method for fault detection and diagnosis are used to identify failure points in IoT [10], [11]. However, similar methods may be developed to detect and diagnose faults in fog nodes.

Without any data backup, IoT data submitted to the fog may be lost when any component of the fog system fails. However, using uncoordinated checkpoints to periodically save the state of fog nodes can improve data recovery and fog reliability [12]. Checkpoints are saved in stable storage, i.e., a repository server that is unaffected by failures. Stable storage is often hosted in dedicated storage infrastructure separate from processing fog nodes [12], [13]. When a failure is detected, repository servers use the most recent checkpoint to restore the state of the lost fog nodes.

A failure within the fog system can create a network overload since the same level of IoT traffic is reaching a hindered fog infrastructure. This may result in the activation of on-demand fog nodes, and data recovery from repository servers. However, current literature does not 1) address the resource allocation of IoT data from repository servers to on-demand fog nodes, nor 2) consider the optimal number, location, or resource quantity of on-demand fog nodes to support a given fog infrastructure.



(a) IoT Hierarchy



(b) Task Workflow

Figure 1: Each physical IoT device has one or more running applications. Each application may execute multiple requests over time. Each request has one or more tasks, arranged in a workflow which defines the dependency relationships between tasks.

D. Summary of Limitations and Strengths

We identify and address two limitations of current contributions: (1) design & dimensioning techniques are sensitive to proper IoT traffic estimations, and assume a reliable fog network; and (2) resource offloading techniques assume the necessary dynamic fog resources are available to support an overloaded fog system. We adopt dynamic network design techniques to design & dimension a fog infrastructure that supports fluctuating IoT traffic and fog node failures. This fog infrastructure is therefore reliable, and robust to dynamic network conditions.

III. IoT-Fog Network

We consider an infrastructure of fog microdata centres to support and service nearby IoT applications. We mathematically define the components of the IoT-Fog network, as well as the interactions between the two layers. When allocating IoT requests to fog resources, we consider resources to include processing ($\texttt{CPU}$), memory ($\texttt{RAM}$), and long-term storage ($\texttt{STR}$). We define R to be the set of resources such that $R = \{\texttt{CPU}, \texttt{RAM}, \texttt{STR}\}$.

A. IoT Virtual Network

The IoT layer is composed of heterogeneous IoT devices, each with a different computing capability, location, and communication range. Each IoT device may run one or several IoT *applications*. Each IoT application may launch several IoT *requests* on an independent request schedule. Each IoT request is composed of one or more IoT *tasks*. Each IoT task has individual resource, bandwidth, and latency requirements for task processing. The IoT hierarchy is shown in Fig. 1. Any property held by a parent element is inherited by the child, e.g., a task's location.

Let N be the set of all IoT requests, T be the set of all IoT tasks, and E be the set of all directed *virtual task dependency links* (VTDLs). Each VTDL links two immediately successive tasks $(t, t') \in T$. We define $T^n \subseteq T$ to be the set of all IoT tasks that belong to IoT request $n \in N$. Likewise, we define E^n to be the set of all VTDL that belong to n. We represent each IoT request n as a Task Dependency Graph (TDG) $J^n = (T^n, E^n)$, and define the set of all directed paths of J^n as Ω^n.

The *workflow* of tasks T^n defines the dependencies between tasks of request n as shown in Fig. 1b. If $t \in T^n$ is *dependent* on $t' \in T^n$, then t' must be processed in its entirety before t may begin to be processed. If t' and t are executed in immediate succession, then we define $next(t') = t$ and $pred(t) = t'$ meaning t is *immediately dependent* on t'. If $t \in T^n$ is the source of a workflow path, then t is not dependent on any task, hence $pred(t) = \emptyset$. Similarly, if $t \in T^n$ is the target of a workflow path, then there does not exist a task that is dependent on t, hence $next(t) = \emptyset$.

For $t, t' \in T^n$ such that t' is dependent on t, we define the set of directed paths from t to t' as $\Omega^n_{t \nrightarrow t'} \subseteq \Omega^n$. For the latency sensitivity of requests, we are interested in the longest possible path. We define the set of end-to-end paths in Ω^n $n \in N$ as

$$\bar{\Omega}^n = \{\Omega^n_{t \nrightarrow t'} \mid pred(t) = \emptyset,\ next(t') = \emptyset,\ t, t' \in T^n\}. \quad (1)$$

For some path $\omega \in \bar{\Omega}^n$, we define ω^{T} and ω^{E} to respectively be the sequence of tasks and of VTDLs in ω. That is,

$$\omega^{\mathrm{E}} = \{e_{i,j} = (t_i, t_j) : next(t_i) = t_j \mid t_i, t_j \in \omega^{\mathrm{T}}\}. \quad (2)$$

B. Fog Physical Infrastructure

Each fog node may be static or dynamic in terms of its availability. Dynamic fog nodes can be viewed as being on *stand-by*, or being available *on-demand*. When IoT traffic increases considerably, or when there is a node failure in the fog infrastructure, a dynamic fog node will activate to support the overloaded system. Let $\hat{F}$ be the set of fixed static fog nodes, and $\widetilde{F}$ be the set of candidate dynamic fog nodes such that $|\widetilde{F}| < |\hat{F}|$. The set of all fog nodes is therefore $F = \hat{F} \cup \widetilde{F}$.

Cloud servers can be useful in processing non-latency sensitive IoT requests, or providing long-term storage for requests processed by fog. We define a *simple router* as a network node with routing capabilities, but no fog resource capabilities. An IoT request may access fog and cloud resources through an *access point* such as a simple router or another fog node. A path between two fog nodes is often not direct, but composed of several simple routers. However, the routing delay incurred by the simple routers within an edge network is minimal, and dominated by the processing delay of fog nodes. Therefore, we simplify the delay formulation by omitting simple routers when discussing network paths.

Within the fog infrastructure, we define a link as a bidirectional connection between two adjacent fog nodes $f, f' \in F$. If at least one incident node is a candidate dynamic fog node $f \in \widetilde{F}$, then the link is a dynamic link; otherwise, it is fixed. Let $\hat{L}$ be the set of fixed fog links, and $\tilde{L}$ be the set of dynamic fog links, and $L = \hat{L} \cup \tilde{L}$ be the set of all fog links.

Between two fog nodes, if no direct link exists between them, then we assume there exists some path consisting of direct links that connect both nodes. Let Π be the set of all paths between fog nodes in F. For a path $\pi \in \Pi$, let $\pi^{\mathrm{L}} = \{\ell_1, \ell_2, \dots\}$ be the sequence of fog links in path π. We define $\Pi_{f \nrightarrow f'} \subseteq \Pi$ to be the set of all possible paths in Π from $f \in F$ to $f' \in F$ and $\Pi^{\ell} \subseteq \Pi$ to be the set of all paths in Π that include the link $\ell \in L$.

We denote the set of all paths with a fog node $f \in F$ as the source or target respectively as

$$\Pi_f^{\text{OUT}} = \bigcup_{f' \in F} \Pi_{f \to f'} \tag{3}$$

$$\Pi_f^{\text{IN}} = \bigcup_{f' \in F} \Pi_{f' \to f} \tag{4}$$

$$\Pi_f = \Pi_f^{\text{IN}} \cup \Pi_f^{\text{OUT}}. \tag{5}$$

For simplicity, we define $\text{D} = \{\text{IN}, \text{OUT}\}$ as the set of path directions to/from a fog node.

We define the set of paths between static fog nodes as

$$\hat{\Pi} = \bigcup_{f \in \hat{F}} \bigcup_{f' \in \hat{F}} \Pi_{f \to f'}. \tag{6}$$

Likewise, we define the set of paths with exactly one dynamic fog node at its source, target or both as

$$\widetilde{\Pi} = \bigcup_{\tilde{f} \in \widetilde{F}} \left[\Pi_{\tilde{f} \to \tilde{f}} \bigcup_{f \in \hat{F}} \left(\Pi_{f \to \tilde{f}} \cup \Pi_{\tilde{f} \to f} \right) \right]. \tag{7}$$

Various path notations can be overlapped to define further path subsets; e.g. $\hat{\Pi}^{\ell}_{f \to f'} = \hat{\Pi} \cap \Pi_{f \to f'} \cap \Pi^{\ell}$. The set parameters of static and dynamic fog nodes and paths are summarized in Table I.

C. Fog Design & Dimensioning

Our previous work [8] designs & dimensions a static fog infrastructure based on estimated IoT traffic. The method used to design & dimension a fog infrastructure is to create a surjective mapping of each IoT task to a fog node, and each VTDL to a fog path. If $\hat{F}$ and $\hat{L}$ are the sets of designed & dimensioned fog nodes and links, then this paper aims to enhance the static fog infrastructure with a dynamic extension. The solution to this problem will yield a valid node mapping $\mathscr{f} : T \mapsto \hat{F}$ for each IoT task, and a valid path mapping $\varpi : E \mapsto \hat{\Pi}$ for each VTDL. These mappings ensure that each task and each VTDL is mapped to exactly one fog node and fog path respectively. These functions are described in Table II. From these mappings, we define binary variables $\hat{x}_f^t$ and $\hat{x}_\pi^e$ such that

$$\hat{x}_f^t = \mathbb{1} \left[\mathscr{f}(t) = f \right] \tag{8}$$

$$\hat{x}_\pi^e = \mathbb{1} \left[\varpi(e) = \pi \right]. \tag{9}$$

It is assumed the fog infrastructure has no unused fog nodes and no redundant requests. That is, $\sum_{t \in T} \hat{x}_f^t = 1$ for each $f \in \hat{F}$ and $\sum_{e \in E} \hat{x}_\pi^e = 1$ for each $\pi \in \hat{\Pi}$.

Each static fog node $f \in \hat{F}$ has a fixed set of associated tasks T_f, and each path $\pi \in \hat{\Pi}_f$ has a fixed set of associated VTDLs E_π. Each IoT task $t \in T_f$ has resource requirement p_t^r of $r \in R$, and bandwidth requirement b_t. The bandwidth requirement of a VTDL $e \in E_\pi$ such that $e = (t, t')$ is $b_e = b_t$.

Let $\hat{z}_f^r$ and $\hat{q}_\pi$ represent the quantity of reallocated resource and bandwidth when a fog node f fails, such that

$$\hat{z}_f^r = \sum_{t \in T_f} p_t^r; \quad r \in R \tag{10}$$

$$\hat{q}_\pi = \sum_{e \in E_\pi} b_e. \tag{11}$$

Table I: Parameters for the fog physical infrastructure.

Parameter	Description
$\hat{F}, \widetilde{F}$	Set of static and dynamic fog nodes respectively.
$\hat{\Pi}, \widetilde{\Pi}$	Set of static and dynamic fog paths respectively.
Π_f^{d}	Set of paths in direction $\mathrm{d} \in \mathrm{D}$ to/from $f \in \hat{F} \cup \widetilde{F}$.
$\bar{P}_f^r$	Maximum resource capacity $r \in R$ in $f \in \hat{F} \cup \widetilde{F}$.
$\bar{B}_\ell$	Maximum bandwidth capacity in $\ell \in \hat{L} \cup \tilde{L}$.
ξ_f	Failure rate of $f \in \hat{F}$.

Table II: Mapping functions of the IoT-fog network.

Function	Mapping	Description
$\mathscr{f}(t)$	$T \mapsto \hat{F}$	Maps IoT task to a static fog node.
$\varpi(e)$	$E \mapsto \Pi$	Maps a virtual task dependency link to a physical fog path.
$\tilde{\mathscr{f}}(f)$	$\hat{F} \mapsto \widetilde{F}$	Maps a fog node with a backup dynamic fog node.
$\widetilde{\varpi}^{\mathrm{d}}(\pi)$	$\hat{\Pi} \mapsto \widetilde{\Pi}$	Maps a virtual task dependency link to an alternate dynamic fog path in direction $\mathrm{d} \in \mathrm{D}$.

IV. Network Latency

The IoT landscape is composed of several types of devices with different processing request patterns. Certain devices transmit data to cloud on a periodic schedule, some transmit data frequently with stochastic inter-transmission times, and others transmit on an ad hoc basis. In complex systems where data presents burst behaviour, e.g. Smart Cities, IoT traffic can be modelled by a Poisson distribution under reasonable aggregation assumptions [23], [24]. Therefore, we model IoT request arrival and inter-arrival to the fog network by a Poisson and Exponential distribution respectively.

A. IoT Traffic

The set N is obtained from historic or estimated data of IoT requests. Though IoT devices may have varied distributions of request transmission, we assume the entire IoT environment exhibits IoT request transmissions that follow a Poisson Process [23]. The total IoT task arrival rate to each fog node is

$$\hat{\lambda}_f = \sum_{n \in N} \sum_{t \in T^n} \hat{x}_f^t \cdot \hat{\lambda}_n; \quad f \in \hat{F} \tag{12}$$

where $\hat{\lambda}_n$ is the estimated request transmission rate of $n \in N$ per unit time.

B. Transmission

Each IoT task $t \in T$ has bandwidth requirement b_t bytes/second, request (upload) byte size s_t^+, and response (download) request byte size s_t^-. We define the upload and download transmission times, respectively, as

$$\nu_t^+ = \frac{b_t}{s_t^+} \tag{13}$$

$$\nu_t^- = \frac{b_t}{s_t^-}. \tag{14}$$

C. Processing Time

The quantity and frequency of computing resources $\hat{z}_f^{\mathrm{CPU}}$ determine how quickly a fog node can process an IoT request. Indeed, a faster processing will yield quicker service. We define the local processing resources of task $t \in T$ as z_t^{T} and the local processing time as ζ_t^{T}. If z_t^{T} is represented in gigahertz, then the number of cycles required to process task $t \in T$ is

$$h_t = z_t^{\mathrm{T}} \cdot \zeta_t^{\mathrm{T}}. \tag{15}$$

The processing time of IoT task $t \in T$ on fog node $f \in \hat{F}$ is

$$\zeta_{t,f}^{\mathrm{F}} = \frac{h_t}{\hat{z}_f^{\mathrm{CPU}}}. \tag{16}$$

The expected processing time by a fog node of any IoT task is

$$\frac{1}{\hat{\mu}_f} = \mathbb{E}_f[\zeta] = \frac{\sum_{t\in T} \hat{x}_f^t \cdot \zeta_{t,f}^{\mathrm{F}}}{\sum_{t\in T} \hat{x}_f^t}; \quad f \in \hat{F} \tag{17}$$

$$\frac{1}{\hat{\varsigma}_f^2} = \mathbb{E}_f[\zeta^2] = \frac{\sum_{t\in T} \hat{x}_f^t \cdot (\zeta_{t,f}^{\mathrm{F}})^2}{\sum_{t\in T} \hat{x}_f^t}; \quad f \in \hat{F}. \tag{18}$$

D. Congestion

At fog node $f \in \hat{F}$, the IoT task arrival rate $\hat{\lambda}_f$ (12) follows a Poisson Process [23], with expected processing time $1/\hat{\mu}_f$ (17). Therefore, congestion on a single link towards f can be modelled as an $M/G/1$ queue. We determine the average waiting time [25] by the function $\eta : \mathbb{R}^2 \mapsto \mathbb{R}$ to be

$$\eta(\lambda, \mu, \varsigma^2) = \frac{\lambda}{2\varsigma^2(1 - \frac{\lambda}{\mu})}. \tag{19}$$

For processes arriving to static fog node $f \in \hat{F}$, we define $\hat{\eta}_f = \eta(\hat{\lambda}_f, \hat{\mu}_f, \hat{\varsigma}_f^2)$.

E. End-to-end latency

Consider a path $\omega \in \bar{\Omega}^n$. We enumerate the tasks such that $\omega^{\mathrm{T}} = \{t_1, t_2, \dots\}$, and express the fog node mapping as $f_i = \not{f}(t_i)$. We then define the end-to-end latency of IoT request execution $n \in N$ over task workflow $\omega \in \bar{\Omega}^n$ as

$$\hat{\Delta}_\omega^n = \nu_{t_1}^+ + \nu_{t_{|\pi^{\mathrm{T}}|}}^- + \sum_{i=1}^{|\omega^{\mathrm{T}}|} \left(\zeta_{t_i,f_i}^{\mathrm{F}} + \hat{\eta}_{f_i}\right). \tag{20}$$

If τ_n is the latency threshold of request n, then $\hat{\Delta}_\omega^n \le \tau_n$, and $\bar{\Delta}_\omega^n = \tau_n - \hat{\Delta}_\omega^n$ is the remaining latency threshold over n.

V. Design & Dimensioning of Reliable Fog Network

The dimensioned design of the fog infrastructure is optimized to support the estimated IoT traffic. Therefore, a larger estimate of IoT traffic may be selected to create a reliable network. However, in the event a single fog node fails, surrounding fog nodes may not be able to host the failed fog node's IoT processes due to resource constraints. This issue can be alleviated using dynamic fog nodes of equal or larger resource capacity. These on-demand nodes become active when a fog node fails. Since a dynamic fog node is not executing any processes upon waking, it can host all of the failed fog node's computations. When all fog nodes are operating properly and IoT traffic is manageable, the dynamic fog node may: (1) power down, thus saving energy and operation cost, or (2) assist in servicing IoT requests of certain priority, thus decreasing overall latency.

Table III: Decision variables for MILP and CG formulation.

Decision Variables	Domain	Description
$y_{\tilde{f}}$	$\{0,1\}$	Design of fog node $\tilde{f} \in \tilde{F}$.
$w_{\tilde{\ell}}$	$\{0,1\}$	Design of link $\tilde{\ell} \in \tilde{L}$.
$z_{\tilde{f}}^r$	$\left[0, \bar{P}_{\tilde{f}}^r\right]$	Dimension of resources $r \in R$ allocated to Fog node $\tilde{f} \in \tilde{F}$.
$q_{\tilde{\ell}}$	$\left[0, \bar{B}_{\tilde{\ell}}\right]$	Dimension of bandwidth allocated to Fog link $\tilde{\ell} \in \tilde{L}$.
$x_{f,\tilde{f}}$	$\{0,1\}$	Dynamic association of $f \in \hat{F}$ to $\tilde{f} \in \tilde{F}$.
$x_{\pi,\tilde{\pi}}^{\mathrm{d}}$	$\{0,1\}$	Dynamic association of $\pi \in \hat{\Pi}$ to $\tilde{\pi} \in \widetilde{\Pi}$ in direction $\mathrm{d} \in \mathrm{D}$.
λ_ψ	$\{0,1\}$	Selection of DFAC $\psi \in \Psi$.

We propose the reliable design & dimensioning of dynamic fog nodes to support fog node failures in a static fog infrastructure. If a fog node fails and a nearby fog node has ample resources, a migration of resources may occur to the active fog node. In the following formulation, we are interested in the worst-case scenario where IoT traffic uses the full capacity of fog, and subsequent fog node failures are experienced.

Subsection V-A defines the assumptions of the model. Subsection V-B describes the decision variables and constraints related to the association of static and dynamic fog nodes and paths. Subsection V-C describes the decision variables and constraints related to the design & dimensioning of dynamic fog nodes and links. Subsection V-D details the objective function for reliable design & dimensioning. Subsection V-E and V-F define the remaining constraints related to reliability, resource requirements, and latency requirements.

A. Assumptions

When a static fog node fails and the dynamic fog node activates, a repository node migrates the most recent data checkpoint to the dynamic fog node. We assume this process is automatically triggered upon failure, and does not produce significant interruption of fog services to IoT. For model simplicity, we assume dynamic fog nodes are dedicated servers that cannot fail. The dynamic fog nodes incur energy costs while they are active. We assume there are no energy costs associated with an inactive dynamic fog node.

Our model focuses on the location and resource placement of dynamic fog nodes towards the implementation of a reliable fog extension. The proposed reliable fog model does not consider any time dependent aspects of operation such as energy costs or data migration since these are outside the scope of design & dimensioning. Furthermore, the placement of stable storage [12], either in a dedicated server or distributed among nearby fog nodes, warrants dedicated investigation & experimentation, and is therefore excluded from this model.

B. Fog Association

We define the mapping $\tilde{\not\ell}(f) : \hat{F} \mapsto \widetilde{F}$ to associate a static fog node with exactly one backup dynamic fog node. When a fog node $f \in \hat{F}$ fails such that $\tilde{\not\ell}(f) = \tilde{f}$, then all tasks currently assigned to f, i.e., $t \in T_f$, must be reallocated to $\tilde{f}$. We introduce a binary decision variable $x_{f,\tilde{f}}$ such that

$$x_{f,\tilde{f}} = \mathbb{1}\left[\tilde{\not\ell}(f) = \tilde{f}\right]; \quad f \in \hat{F} \tag{21}$$

$$\implies \sum_{\tilde{f} \in \widetilde{F}} x_{f\tilde{f}} = 1; \quad f \in \hat{F}. \tag{22}$$

In the event of a fog node failure, all static paths connected to the failed fog node are replaced with alternate paths connected to the dynamic fog node. The target of an *in*-path is a re-associated fog node, and the source of an *out*-path is a re-associated fog node. We define the mappings $\widetilde{\varpi}^{\text{IN}} : \Pi \mapsto \widetilde{\Pi}$ and $\widetilde{\varpi}^{\text{OUT}} : \Pi \mapsto \widetilde{\Pi}$ to respectively associate a path $\pi \in \Pi$ to alternate dynamic paths $\tilde{\pi}^{\text{IN}}, \tilde{\pi}^{\text{OUT}} \in \widetilde{\Pi}$. We introduce binary decision variables $x^{\text{IN}}_{\pi,\tilde{\pi}}$ and $x^{\text{OUT}}_{\pi,\tilde{\pi}}$ such that

$$x^{\text{d}}_{\pi,\tilde{\pi}} = \mathbb{1}\left[\widetilde{\varpi}^{\text{d}}(\pi) = \tilde{\pi}\right]; \quad \pi \in \hat{\Pi}^{\text{d}},\ \text{d} \in \text{D} \tag{23}$$

$$\implies \sum_{\tilde{\pi} \in \widetilde{\Pi}^{\text{d}}} x^{\text{d}}_{\pi,\tilde{\pi}} = 1; \quad \pi \in \hat{\Pi}^{\text{d}},\ \text{d} \in \text{D}. \tag{24}$$

Equations (22) and (24) are, respectively, the node association and path association constraints of the set partitioning problem [14]. Finally, the path association of a static path is bounded below by the appropriate fog node association. In other words,

$$x_{f,\tilde{f}} \leq \sum_{\tilde{\pi} \in \widetilde{\Pi}^{\text{d}}_{\tilde{f}}} x^{\text{d}}_{\pi,\tilde{\pi}};\ \pi \in \hat{\Pi}^{\text{d}}_{f},\ f \in \hat{F},\ \tilde{f} \in \widetilde{F},\ \text{d} \in \text{D}. \tag{25}$$

We restrict the possible fog associations to permissible mappings only. A fog node or path mapping may not be possible due to distance, organizational ownership, or mismatching path endpoints. Let $\tilde{Q}_f$ be the set of all possible dynamic fog nodes $\tilde{f} \in \widetilde{F}$ that can be associated with $f \in \hat{F}$, and let $\tilde{Q}_\pi$ is the set of all possible alternate paths $\tilde{\pi} \in \widetilde{\Pi}$ that can be associated with $\pi \in \Pi$.

$$x_{f,\tilde{f}} = 0; \quad \forall (f, \tilde{f}) \in \hat{F} \times (\widetilde{F} \setminus \tilde{Q}_f) \tag{26}$$

$$x_{\pi,\tilde{\pi}} = 0; \quad \forall (\pi, \tilde{\pi}) \in \hat{\Pi} \times (\widetilde{\Pi} \setminus \tilde{Q}_\pi) \tag{27}$$

Constraints (26) and (27), respectively, invalidate any impermissible fog node or path mapping.

C. Design & Dimensioning

We define four decision variables for the design and dimensioning of the fog infrastructure. The continuous dimensioning of resource $r \in R$ of dynamic fog node $\tilde{f} \in \widetilde{F}$ is denoted $z^r_{\tilde{f}} \in \left[0, \bar{P}^r_{\tilde{f}}\right]$. The continuous dimensioning of bandwidth in fog link $\tilde{\ell} \in \tilde{L}$ is denoted $q_{\tilde{\ell}} \in \left[0, \bar{B}_{\tilde{\ell}}\right]$. The associated binary design variables are denoted $y_{\tilde{f}}$ and $w_{\tilde{\ell}}$ such that

$$y_{\tilde{f}} = \left[\sum_{r \in R} z^r_{\tilde{f}} > 0\right]; \quad \tilde{f} \in \widetilde{F} \tag{28}$$

$$w_{\tilde{\ell}} = \left[q_{\tilde{\ell}} > 0\right]; \quad \tilde{\ell} \in \tilde{L}. \tag{29}$$

If a design variable is positive, we say that node/link is *selected.* The domain and descriptions of design and dimensioning variables are summarized in Table III.

We define the relationship between design & dimensioning decision variables through capacity constraints. Let $W_{\tilde{f}}$ be the set of links incident to $\tilde{f} \in \widetilde{F}$, and let $W^{\text{MAX}}_{\tilde{f}}$ be the maximum number of selected links incident to $\tilde{f}$.

$$z^r_{\tilde{f}} \leq \bar{P}^r_{\tilde{f}} \cdot y_{\tilde{f}}; \quad \tilde{f} \in \widetilde{F},\ r \in R \tag{30}$$

$$q_{\tilde{\ell}} \leq \bar{B}_{\tilde{\ell}} \cdot w_{\tilde{\ell}}; \quad \tilde{\ell} \in \tilde{L} \tag{31}$$

$$\sum_{\tilde{\ell} \in W_{\tilde{f}}} w_{\tilde{\ell}} \leq W^{\text{MAX}}_{\tilde{f}} \cdot y_{\tilde{f}}; \quad \tilde{f} \in \widetilde{F} \tag{32}$$

$$\sum_{\tilde{f} \in \widetilde{F}} y_{\tilde{f}} \leq Y_{\text{MAX}}, \tag{33}$$

Constraint (30) enforces the selection of a fog node that is dimensioned a positive resource amount. Constraint (31) enforces the selection of a fog link that is dimensioned a positive bandwidth amount. Constraint (32) defines the relationship between an selected fog node and incident links. Finally, (33) defines the maximum number of selected fog nodes.

D. Objective function

We formulate the problem as an MILP. For each dynamic fog node and path mapping, we calculate the difference in cost of the alternate mapping from the original, i.e., the *additional* mapping cost. Furthermore, we calculate the implementation cost of each dynamic fog node and fog link. Together, these represent the design and dimensioning costs of the set partitioning problem. Therefore, this problem is formally defined as follows. *We seek the most efficient mappings* $\{\tilde{\not\ell}(f) \mid \forall f \in \hat{F}\}$, *and* $\{\widetilde{\varpi}^{\text{d}}(\pi) \mid \forall \pi \in \hat{\Pi},\ \text{d} \in \text{D}\}$ *that minimize the design costs of dynamic fog association.*

$$\begin{aligned}
\min & \sum_{\tilde{f} \in \widetilde{F}} V_{\tilde{f}} \cdot y_{\tilde{f}} + \sum_{\tilde{\ell} \in \tilde{L}} U_{\tilde{\ell}} \cdot w_{\tilde{\ell}} \\
& + \sum_{\tilde{f} \in \widetilde{F}} \sum_{r \in R} v^r_{\tilde{f}} \cdot z^r_{\tilde{f}} + \sum_{\tilde{\ell} \in \tilde{L}} u_{\tilde{\ell}} \cdot q_{\tilde{\ell}} \\
& + \sum_{f \in F} \sum_{\tilde{f} \in \widetilde{F}} \sum_{r \in R} \hat{z}^r_f \cdot \left(c^r_{\tilde{f}} - c^r_{\not\ell(t)}\right) \cdot x_{f,\tilde{f}} \\
& + \sum_{\pi \in \hat{\Pi}} \sum_{\tilde{\pi} \in \widetilde{\Pi}} \sum_{\text{d} \in \text{D}} \hat{q}_\pi \cdot \left(\sum_{\ell \in \tilde{\pi}^{\text{L}}} c_\ell - \sum_{\ell \in \pi^{\text{L}}} c_\ell\right) \cdot x^{\text{d}}_{\pi,\tilde{\pi}}
\end{aligned} \tag{34}$$

The objective function is divided into four sections for the joint optimization of various facets of dynamic fog design & dimensioning. The first line expresses the capital expenditure costs of implementing resources and bandwidth into dynamic

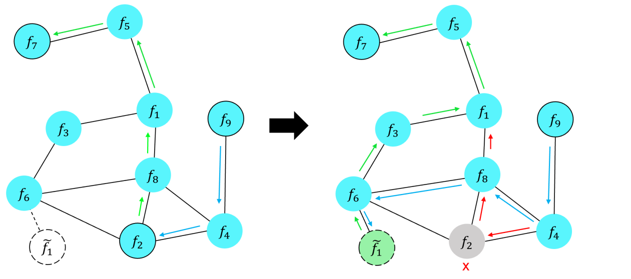


Figure 2: When a fog node $f \in \hat{F}$ fails, such that $\not{f}(f) = \tilde{f}$, then $\tilde{f}$ becomes active and all data allocated to f is reallocated to $\tilde{f}$ along several possible paths.

fog nodes and links respectively. The second line expresses the unit costs of implementing resources and bandwidth into dynamic fog nodes and links respectively. The third line expresses the cost of associating resources from a failed node to an associated dynamic node. The fourth line expresses the unit costs of associating a static fog path to an associated dynamic path. Since the resources are no longer executed in the failed node, the original resource amounts are subtracted from the failed node and added to the dynamic node with a new unit price. Similarly, the last term expresses the bandwidth costs associated with rerouting IoT request data from the failed node to the dynamic node.

E. Reliability constraint

Suppose $f \in \hat{F}$ has a *failure rate* $\xi_f \in (0, 1)$, meaning fog node f has a probability ξ_f of being in a failed state at any given time[1]. Consider a cluster of static fog nodes associated with dynamic fog node $\tilde{f} \in \widetilde{F}$, and denote the worst-case failure rate among the fog nodes as $\xi = \max_{f \in \hat{F}} \xi_f$. Therefore, the probability of experiencing at most one failure for nodes associated with $\tilde{f}$ is

$$\begin{aligned} &Pr(\text{at most one failure}; \tilde{f} \in \widetilde{F}) \\ =&\prod_{f \in \hat{F}} (1-\xi_f)^{x_{f,\tilde{f}}} + \sum_{f \in \hat{F}} x_{f,\tilde{f}} \xi_f \Big[\prod_{f' \in \hat{F} \setminus f} (1-\xi_{f'})^{x_{f',\tilde{f}}} \Big] \\ \geq& (1-\xi)^{n_{\tilde{f}}} + n_{\tilde{f}} \cdot \xi (1-\xi)^{(n_{\tilde{f}}-1)} \qquad (35) \\ =&\Xi(n_{\tilde{f}}; \xi), \end{aligned}$$

where $n_{\tilde{f}} = \sum_{f \in \hat{F}} x_{f,\tilde{f}}$.

To guarantee network reliability, we require the probability of at most one failure in any cluster of fog nodes associated with a dynamic fog node to be at least $\alpha \in (0, 1)$, assuming a worst-case failure rate. In other words,

$$\Xi(n_{\tilde{f}}; \xi) \geq \alpha^{y_{\tilde{f}}}; \quad \tilde{f} \in \widetilde{F}. \tag{36}$$

For high network robustness, we choose α to be close to 1.

In order to linearize failure constraint (35), we consider only the probability that there exists no failure in a cluster. We define α_0 as the probability threshold that there does not exist any fog node failure associated with $\tilde{f}$, such that

$$\alpha_0 = \min_{n \in \mathbb{Z}^{\geq 2}} \Big\{ (1-\xi)^n \mid \Xi(n; \xi) \geq \alpha \Big\}. \tag{37}$$

[1]In practice, we discourage the use of fog nodes with high failure rate.

The linearized failure constraint becomes

$$\begin{aligned} &\prod_{f \in \hat{F}} (1-\xi_f)^{x_{f,\tilde{f}}} \geq \alpha_0^{y_{\tilde{f}}} \\ \iff & \sum_{f \in \hat{F}} x_{f,\tilde{f}} \cdot \log(1-\xi_f) \geq y_{\tilde{f}} \cdot \log \alpha_0; \quad \tilde{f} \in \widetilde{F}. \end{aligned} \tag{38}$$

F. Service reallocation constraints

When all processes from a failed fog node are redirected to a dynamic fog node, we ensure the resource and bandwidth capacities of the fog network, and the latency requirements of the IoT requests are respected. We define the following constraints.

$$x_{f,\tilde{f}} \cdot \hat{z}^r_f \ \leq \ z^r_{\tilde{f}}; \quad (f, \tilde{f}) \in \hat{F} \times \widetilde{F}, r \in R \tag{39}$$

$$x^{\mathrm{d}}_{\pi,\tilde{\pi}} \cdot \hat{q}_\pi \leq q_{\ell'}; \quad (\pi, \tilde{\pi}) \in \hat{\Pi} \times \widetilde{\Pi}, \ell' \in \tilde{\pi}^{\mathrm{L}}, \ \mathrm{d} \in \mathrm{D} \tag{40}$$

$$\Delta^n_{\omega;\tilde{f}} \leq \bar{\Delta}^n_\omega \cdot y_{\tilde{f}}; \quad n \in N, \ \omega \in \bar{\Omega}^n, \ \tilde{f} \in \widetilde{F}. \tag{41}$$

where

$$\begin{aligned} \Delta^n_{\omega;\tilde{f}} =& \sum_{i=1}^{|\omega^{\mathrm{T}}|} \left(\frac{1}{\tilde{\mu}_{f,\tilde{f}}} - \frac{1}{\hat{\mu}_{\not{f}(t_i)}} \right) \cdot x_{\not{f}(t_i),\tilde{f}} \\ &+ \sum_{i=1}^{|\omega^{\mathrm{T}}|} \left(\tilde{\eta}_{\not{f}(t_i),\tilde{f}} - \hat{\eta}_{\not{f}(t_i)} \right) \cdot x_{\not{f}(t_i),\tilde{f}}. \end{aligned} \tag{42}$$

$$\tilde{\eta}_{f,\tilde{f}} = \eta(\hat{\lambda}_f, \tilde{\mu}_{f,\tilde{f}}, \tilde{\varsigma}^2_{f,\tilde{f}}) \tag{43}$$

$$\tilde{\mu}_{f,\tilde{f}} = \frac{\hat{\mu}_f \cdot z^{\mathrm{CPU}}_{\tilde{f}}}{\hat{z}^{\mathrm{CPU}}_f} \tag{44}$$

$$\tilde{\varsigma}^2_{f,\tilde{f}} = \frac{\hat{\varsigma}^2_f \cdot z^{\mathrm{CPU}}_{\tilde{f}}}{\hat{z}^{\mathrm{CPU}}_f}. \tag{45}$$

Constraint (39) expresses the minimum resource dimensioning a dynamic fog node may have. Constraint (40) expresses the bandwidth constraints of rerouting a task to a new physical path. For link $\ell \in L$, the constraint defines the maximum bandwidth as a) the remaining bandwidth on a static link, or b) the bandwidth dimensioning of a dynamic link. In other words,

$$q_\ell = \begin{cases} \bar{B}_\ell & \text{if } \ell \in \hat{L} \\ q_{\tilde{\ell}} & \text{if } \ell \in \tilde{L}. \end{cases} \tag{46}$$

Similarly, (41) requires the additional latency generated from rerouting to a dynamic fog node to be constrained by the remaining latency threshold. The latency calculation (42) has two terms describing the latency incurred from using a dynamic path. These represent the additional latency incurred by processing time and congestion respectively.

VI. Linearization of Reliable Model

A. Computing resource constraint

Dimensioning the computing resources of a fog node will also affect processing time, and therefore the congestion of that fog node. Indeed, higher computing resources will allow a fog node to process requests faster. We introduce an array of decision variables $\breve{z}_{\tilde{f}} = [\breve{z}^0_{\tilde{f}}, \ldots, \breve{z}^Z_{\tilde{f}}]$, $\breve{z}^i_{\tilde{f}} \in \{0, 1\}$, for a large integer Z. Let $s_{\tilde{f}} = \bar{P}^{\texttt{CPU}}_{\tilde{f}}/Z$, where $\bar{P}^{\texttt{CPU}}_{\tilde{f}}$ is the maximum computing resource capacity of dynamic fog node $\tilde{f}$. The congestion calculation (43) is a non-linear term in $z^{\texttt{CPU}}_{\tilde{f}}$ (44). Hence, we reformulate $\tilde{\eta}_{f,\tilde{f}}$ as a continuous piecewise decision variable.

$$\sum_{i=0}^{Z} i \cdot s_{\tilde{f}} \cdot \breve{z}^{i}_{\tilde{f}} = z^{\texttt{CPU}}_{\tilde{f}} \tag{47}$$

$$\sum_{i=0}^{Z} \breve{z}^{i}_{\tilde{f}} = 1. \tag{48}$$

$$\begin{aligned}\tilde{\eta}^{i}_{f,\tilde{f}} &= \eta(\hat{\lambda}_f, \frac{i \cdot s_{\tilde{f}}}{\hat{z}^{\texttt{CPU}}_{f}} \cdot \hat{\mu}_f, \frac{i \cdot s_{\tilde{f}}}{\hat{z}^{\texttt{CPU}}_{f}} \cdot \hat{\varsigma}^2_f) \cdot \breve{z}^{i}_{\tilde{f}}.\\ \implies \tilde{\eta}_{f,\tilde{f}} &= \sum_{i=0}^{Z} \tilde{\eta}^{i}_{f,\tilde{f}}\end{aligned} \tag{49}$$

Constraints (47) and (48) equate the value $z^{\texttt{CPU}}_{\tilde{f}}$ to a single partition variable $z^{i}_{\tilde{f}}$ where the coefficient is a fraction of the maximum computing resource capacity. Constraint (49) assigns $\eta_{\tilde{f}}$ to the same partition variable with congestion coefficients.

For static fog node $f \in \hat{F}$ and dynamic fog node $\tilde{f} \in \widetilde{F}$, components of (42) and (44) include terms $x_{f,\tilde{f}}/z^{\texttt{CPU}}_{\tilde{f}}$ and $\tilde{\eta}_{f,\tilde{f}} \cdot x_{f,\tilde{f}}$. For a function $g(\breve{z}_f) = \sum_{i=0}^{Z} \breve{z}^{i}_{f} \cdot C_i$ with constants $C_1, \ldots, C_Z$, consider the product $g(\breve{z}_{\tilde{f}}) \cdot x_{f,\tilde{f}}$ and reformulate the non-linear terms.

$$g(\breve{z}_{\tilde{f}}) \cdot x_{f,\tilde{f}} = \sum_{i=0}^{Z} x_{f,\tilde{f}} \cdot \breve{z}^{i}_{\tilde{f}} \cdot C_i \tag{50}$$

$$\frac{x_{f,\tilde{f}}}{z^{\texttt{CPU}}_{\tilde{f}}} = \sum_{i=1}^{Z} x_{f,\tilde{f}} \cdot \breve{z}^{i}_{\tilde{f}} \cdot \underbrace{\frac{1}{i \cdot s_{\tilde{f}}}}_{C_i} \tag{51}$$

$$\tilde{\eta}_{f,\tilde{f}} \cdot x_{f,\tilde{f}} = \sum_{i=0}^{Z} x_{f,\tilde{f}} \cdot \breve{z}^{i}_{\tilde{f}} \cdot \underbrace{\tilde{\eta}^{i}_{f,\tilde{f}}}_{C_i} \cdot \tag{52}$$

We observe that it is only necessary to linearize the terms $x_{f,\tilde{f}} \cdot \breve{z}^{i}_{\tilde{f}}$ for $i = 0, \ldots, Z$ which are products of binary variables. Hence, we introduce a new binary variable $\breve{x}^{i}_{f,\tilde{f}} \in \{0, 1\}$ such that

$$\begin{aligned}\breve{x}^{i}_{f,\tilde{f}} &\leq x_{f,\tilde{f}}\\ \breve{x}^{i}_{f,\tilde{f}} &\leq \breve{z}^{i}_{\tilde{f}}\end{aligned} \qquad i = 1, \ldots, Z. \tag{53}$$

Therefore, the processing and congestion terms of (42), respectively, become

$$\frac{x_{f,\tilde{f}}}{z^{\texttt{CPU}}_{\tilde{f}}} = \sum_{i=1}^{Z} \frac{\breve{x}^{i}_{f,\tilde{f}}}{i \cdot s_{\tilde{f}}} \tag{54}$$

$$\tilde{\eta}_{f,\tilde{f}} \cdot x_{f,\tilde{f}} = \sum_{i=0}^{Z} \breve{x}^{i}_{f,\tilde{f}} \cdot \tilde{\eta}^{i}_{f,\tilde{f}}. \tag{55}$$

VII. Column Generation Heuristic for Design & Dimensioning of a Reliable Fog Network

The MILP formulation is intractable for large numbers of static and dynamic fog nodes. Mapping each static fog node to exactly one dynamic fog node yields $|\widetilde{F}|^{|\hat{F}|}$ possible combinations of node mappings. Mapping each static fog path to one dynamic fog path per direction yields $2|\widetilde{\Pi}|^{|\hat{\Pi}|}$ possible combinations of path mappings. Since every dynamic fog link is incident to a dynamic fog node, of which there are $|\widetilde{F}|$, then $|\widetilde{L}| \propto |\widetilde{F}|$. Therefore, the fog association components of fog-RO-MILP are the most restrictive in terms of solution time.

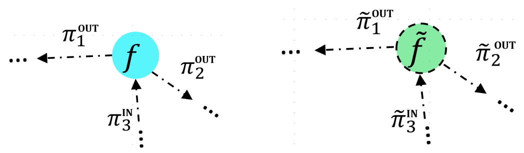

π_1^{OUT}
f
π_2^{OUT}
π_3^{IN}
$\tilde{\pi}_1^{\text{OUT}}$
$\tilde{f}$
$\tilde{\pi}_2^{\text{OUT}}$
$\tilde{\pi}_3^{\text{IN}}$

Figure 3: A DFAC ψ maps a static fog node and all connecting paths to a dynamic node and associated paths.

We propose the use of column generation [15] to increase the scalability of the MILP formulation of the set partitioning problem. We reformulate the fog association variables of fog-RO-MILP in terms of Dynamic Fog Association Configurations (DFACs). DFACs associate each static fog node to exactly one designed dynamic fog node, and each associated static path to at most two directed dynamic paths. In case of a node failure, all IoT traffic from the failed fog node is rerouted to the dynamic node and paths per the appropriate DFAC.

Let Ψ be the set of DFACs. A DFAC is composed of mappings from static fog nodes and paths to dynamic fog nodes and paths respectively. Each DFAC $\psi \in \Psi$ 1) associates a static fog node $f \in \hat{F}$ with a dynamic fog node $\tilde{f} \in \widetilde{F}$, 2a) associates every static path $\pi \in \Pi^{\texttt{IN}}_{f}$ from a node $f' \in \hat{F}$ to f with a dynamic path $\tilde{\pi} \in \widetilde{\Pi}^{\texttt{IN}}_{\tilde{f}}$ from f' to $\tilde{f}$, and 2b) associates a path $\pi \in \Pi^{\texttt{OUT}}_{f}$ from f to f' with a dynamic path $\tilde{\pi} \in \widetilde{\Pi}^{\texttt{OUT}}_{\tilde{f}}$ from $\tilde{f}$ to f'. A DFAC ψ_f that associates f is defined as

$$\widetilde{\boldsymbol{\varpi}}^{\mathrm{d}}_{f} = \{\widetilde{\varpi}^{\mathrm{d}}(\pi) \mid \pi \in \hat{\Pi}^{\mathrm{d}}_{f}\}; \;\; f \in \hat{F} \tag{56}$$

$$\psi_f = \left(\tilde{\mathscr{f}}(f), \widetilde{\boldsymbol{\varpi}}^{\texttt{IN}}_{f}, \widetilde{\boldsymbol{\varpi}}^{\texttt{OUT}}_{f}\right); \;\; f \in \hat{F}. \tag{57}$$

A visual representation of a DFAC is shown in Figure 3. Since each DFAC associates a single fog node f, we wish to select $|\hat{F}|$ DFACs that respect the service constraints of the problem. This corresponds to the Master Problem in our column generation approach. Determining the dynamic fog association of each DFAC corresponds to the Pricing Problem.

A. Master Problem

Let $\Psi^* \subseteq \Psi$ be the subsets of optimal configurations for the fog partitioning solution. Then, the set partitioning problem can be reformulated with respect to λ_ψ such that

$$\lambda_\psi = \mathbb{1}\left[\psi \in \Psi^*\right]; \quad \psi \in \Psi. \tag{58}$$

The fog partitioning portion of fog-RO-MILP can be accordingly reformulated with respect to decision variables λ_ψ. We introduce functions over $\psi \in \Psi$ to represent services given by ψ over aspects of the dynamic fog infrastructure, and define them in Table IV. We present the objective function of the

Table IV: Functions over DFAC ψ for fog-RO-CG-M.

Function	Domain	Range	Description
$a_\psi(f)$	$\Psi \times \hat{F}$	{0,1}	Whether DFAC ψ reroutes data from $f \in \hat{F}$.
$\Theta_\psi(\tilde{f})$	$\Psi\times\widetilde{F}$	$\mathbb{R}^+$	The probability of at most one failure associated with $\tilde{f} \in \widetilde{F}$.
$P^r_\psi(f,\tilde{f})$	$\Psi\times\hat{F}\times\widetilde{F}\times R$	$\mathbb{R}^+$	The quantity of resources $r \in R$ mapped from $f \in \hat{F}$ to $\tilde{f} \in \widetilde{F}$.
$B^{\mathrm{d}}_\psi(\pi,\ell)$	$\Psi\times\mathrm{D}\times\Pi\times L$	$\mathbb{R}^+$	The bandwidth used over link $\ell \in L$ in $\pi \in \Pi$ in direction $\mathrm{d} \in \mathrm{D}$.
$\Delta^{\omega;\tilde{f}}_\psi(n)$	$\Psi\times N\times\bar{\Omega}^n\times\widetilde{F}$	$\mathbb{R}^+$	The additional latency of workflow path $\omega \in \bar{\Omega}^n$, $n \in N$ when the dynamic node $\tilde{f} \in \widetilde{F}$ is used.

Master Problem of the column generation process, known as *fog-RO-CG-M*.

$$\begin{aligned}\min \quad & \sum_{\tilde{f}\in\widetilde{F}} \left(V_{\tilde{f}} + \sum_{r\in R} \bar{\boldsymbol{p}}^r \cdot v^r_{\tilde{f}}\right)\cdot y_{\tilde{f}} \\ & + \sum_{\tilde{\ell}\in\tilde{L}} \left(U_{\tilde{\ell}} + \bar{\boldsymbol{b}}\cdot u_{\tilde{\ell}}\right)\cdot w_{\tilde{\ell}} \qquad (59)\\ & + \sum_{\psi\in\Psi} \mathrm{COST}_\psi\, \lambda_\psi \\ & y_{\tilde{f}}, w_{\tilde{\ell}}, \lambda_\psi \in [0,1] \qquad (60)\end{aligned}$$

where

$$\bar{\boldsymbol{p}}^r = \frac{1}{|\hat{F}|}\sum_{f\in\hat{F}} \hat{z}^r_f \qquad (61)$$

$$\bar{\boldsymbol{b}} = \frac{1}{|\hat{L}|}\sum_{\ell\in\hat{L}} \hat{q}_\ell \qquad (62)$$

$$\begin{aligned}\mathrm{COST}_\psi =& \sum_{f\in\hat{F}}\sum_{\tilde{f}\in\widetilde{F}}\sum_{r\in R} P^r_\psi(f,\tilde{f})\cdot\left(c^r_{\tilde{f}} - c^r_f\right) \\ & + \sum_{\mathrm{d}\in\mathrm{D}}\sum_{\pi\in\Pi}\sum_{\ell\in L} B^{\mathrm{d}}_\psi(\pi,\ell)\cdot\left(\sum_{\tilde{\pi}^\ell} c_\ell - \sum_{\pi^\ell} c_\ell\right). \qquad (63)\end{aligned}$$

The expected resource and bandwidth dimensioning quantities, (61) and (62) respectively, are estimated from the current dimensioning of static fog nodes and links. The DFAC cost (63) is the combined costs of fog node and path association.

Fog-RO-CG-M is divided into fog association and design & dimensioning components. The dynamic fog association problem has far more variables and constraints to consider. Therefore, we apply column generation to these components. Feasible and cost effective fog associations are heavily impacted by dynamic fog design. Therefore, we preserve the design decision variables $y_{\tilde{f}}$ (28) and $w_{\tilde{\ell}}$ (29). We preserve the dimensioning costs by replacing the variables $z^r_{\tilde{f}}$ and $q_{\tilde{\ell}}$ with the expected resource and bandwidth dimensioning quantities. The binary variables $y_{\tilde{f}}$, $w_{\tilde{\ell}}$ and λ_ψ are relaxed to be continuous within [0,1].

1) Pricing constraints: The constraints of fog-RO-CG-M are defined as follows.

$$\sum_{\psi\in\Psi} \lambda_\psi \cdot a_\psi(f) = 1; \quad f\in\hat{F}. \qquad (\varphi_f) \qquad (64)$$

$$\sum_{\psi\in\Psi} \lambda_\psi \cdot \Theta_\psi(\tilde{f}) \geq \log\alpha_0 \cdot y_{\tilde{f}}; \quad \tilde{f}\in\widetilde{F} \qquad (\theta_{\tilde{f}}) \qquad (65)$$

$$\sum_{\psi\in\Psi} \lambda_\psi \cdot P^r_\psi(f,\tilde{f}) \leq \bar{P}^r_{\tilde{f}}\cdot y_{\tilde{f}}; \quad f\in\hat{F},\ \tilde{f}\in F, r\in R \qquad (\rho^r_{f,\tilde{f}}) \qquad (66)$$

$$\sum_{\psi\in\Psi} \lambda_\psi \cdot B^{\mathrm{d}}_\psi(\pi,\ell) \leq \bar{B}_\ell \cdot w_\ell; \quad \ell\in L, \pi\in\hat{\Pi}, \mathrm{d}\in\mathrm{D} \qquad (\beta^{\mathrm{d}}_{\pi,\ell}) \qquad (67)$$

$$\sum_{\psi\in\Psi} \lambda_\psi \cdot \Delta^{\omega;\tilde{f}}_\psi(n) \leq \bar{\Delta}^n_\omega \cdot y_{\tilde{f}}; \quad n\in N, \omega\in\bar{\Omega}^n,\ \tilde{f}\in\widetilde{F} \qquad (\delta^{\omega\tilde{f}}_{\ n}) \qquad (68)$$

The fog-RO-MILP constraints are reformulated in terms of DFACs. Constraint (64) represents the reformulated fog association constraints (22) and (24). Constraint (65) represents the reliability constraint (38). Constraints (66) and (67) respectively represent the resource (39) and bandwidth (40) restrictions of fog-RO-CG-M. Constraint (68) represents the latency constraint (41), where $\Delta^{\omega,\tilde{f}}_\psi$ implicitly uses the expected dimensioning $\bar{\boldsymbol{p}}^{\mathrm{CPU}}_{\tilde{f}}$. The associated link in (67) may be static or dynamic. For static link $\ell\in\hat{L}$, $\bar{B}_\ell$ denotes the remaining bandwidth and $w_\ell = 1$.

The scalars φ_f, $\theta_{\tilde{f}}$, $\rho^r_{f,\tilde{f}}$, $\beta^{\mathrm{d}}_{\pi,\ell}$ and $\delta^{\omega,\tilde{f}}_n$ are the dual variables associated with the corresponding constraint (64) – (68) which are used in solving the Pricing Problem.

2) Capacity constraints: The constraints (30) – (33) from the MILP model are used in fog-RO-CG-M. These constraints express relationship between the design & dimensioning variables per dynamic fog node and link, and the maximum number of designed nodes and links.

B. Pricing Problem Ψ

The Pricing Problem considers the reduced cost of new DFACs to the Master Problem. The Pricing Problem, denoted *fog-RO-CG-P*, is constructed from the dual variables associated with (64) – (68). We are interested in finding a new DFAC ψ to add to the model that will yield a negative reduced cost, which is expressed as follows.

$$\begin{aligned}\overline{\mathrm{COST}}_\psi =& \mathrm{COST}_\psi - \sum_{f\in\hat{F}}\varphi_f\cdot a_\psi(f) - \sum_{\tilde{f}\in\widetilde{F}}\theta_{\tilde{f}}\cdot\Theta_\psi(\tilde{f}) \\ & + \sum_{r\in R}\sum_{f\in\hat{F}}\sum_{\tilde{f}\in\widetilde{F}} \rho^r_{f,\tilde{f}}\cdot P^r_\psi(f,\tilde{f}) \\ & + \sum_{\mathrm{d}\in\mathrm{D}}\sum_{\pi\in\hat{\Pi}}\sum_{\ell\in\tilde{L}} \beta^{\mathrm{d}}_{\pi,\ell}\cdot B^{\mathrm{d}}_\psi(\pi,\ell) \qquad (69)\\ & + \sum_{n\in N}\sum_{\omega\in\bar{\Omega}^n}\sum_{\tilde{f}\in\widetilde{F}} \delta^{\omega;\tilde{f}}_n \cdot \Delta^{\omega;\tilde{f}}_\psi(n).\end{aligned}$$

where COST_ψ is defined by (63).

We express (69) in terms of decision variables (21) and (23), and omit the implied notation of DFAC $\psi \in \Psi$. For $f \in \hat{F}$,

$$a_\psi(f) = \sum_{\tilde{f} \in \widetilde{F}} x_{f,\tilde{f}} + \sum_{\mathrm{d} \in \mathrm{D}} \sum_{\pi \in \hat{\Pi}^{\mathrm{d}}_f} \sum_{\tilde{\pi} \in \widetilde{\Pi}} x^{\mathrm{d}}_{\pi,\tilde{\pi}}. \tag{70}$$

For each dynamic fog node $\tilde{f} \in \widetilde{F}$,

$$\Theta_\psi(\tilde{f}) = \sum_{f \in \hat{F}} x_{f,\tilde{f}} \cdot \log(1 - \xi_f). \tag{71}$$

For each static fog node $f \in \hat{F}$, dynamic fog node $\tilde{f} \in \widetilde{F}$, and resource $f \in R$,

$$P^r_\psi(f, \tilde{f}) = \hat{z}^r_f \cdot x_{f,\tilde{f}} \tag{72}$$

For each path $\pi \in \hat{\Pi}$ and fog link $\ell \in L \setminus \pi^{\mathrm{L}}$,

$$B^{\mathrm{d}}_\psi(\pi, \ell) = \sum_{\tilde{\pi} \in \widetilde{\Pi}^\ell} \sum_{\mathrm{d} \in \mathrm{D}} \hat{q}_\pi \cdot x^{\mathrm{d}}_{\pi,\tilde{\pi}}. \tag{73}$$

For each IoT request $n \in N$, task workflow $\omega \in \bar{\Omega}^n$, and dynamic fog node $\tilde{f} \in \widetilde{F}$,

$$\Delta^{\omega;\tilde{f}}_\psi(n) = \Delta^n_{\omega;\tilde{f}} \tag{74}$$

where $\Delta^n_{\omega;\tilde{f}}$ is defined by (42).

Substituting (70)–(74) into the reduced cost formulation (69) results in the objective function of fog-RO-CG-P. We include the set partitioning constraints (22) and (24). Therefore, fog-RO-CG-P becomes

$$\begin{aligned}
\min = &- \sum_{f \in \hat{F}} \varphi_f \left[\sum_{\tilde{f} \in \widetilde{F}} x_{f,\tilde{f}} + \sum_{\mathrm{d} \in \mathrm{D}} \sum_{\pi \in \hat{\Pi}^{\mathrm{d}}_f} \sum_{\tilde{\pi} \in \widetilde{\Pi}} x^{\mathrm{d}}_{\pi,\tilde{\pi}} \right] \\
&- \sum_{\tilde{f} \in \widetilde{F}} \sum_{f \in \hat{F}} \theta_{\tilde{f}} \log(1 - \xi_f) \cdot x_{f,\tilde{f}} \\
&+ \sum_{f \in \hat{F}} \sum_{\tilde{f} \in \widetilde{F}} \sum_{r \in R} \left(c^r_{\tilde{f}} - c^r_f + \rho^r_{f,\tilde{f}} \right) \cdot \hat{z}^r_f \cdot x_{f,\tilde{f}} \\
&+ \sum_{\pi \in \hat{\Pi}} \sum_{\ell \in L \setminus \pi^{\mathrm{L}}} \sum_{\tilde{\pi} \in \widetilde{\Pi}^\ell} \sum_{\mathrm{d} \in \mathrm{D}} \bar{\beta}_\pi \cdot \hat{q}_\pi \cdot x^{\mathrm{d}}_{\pi,\tilde{\pi}} \\
&+ \sum_{n \in N} \sum_{\omega \in \bar{\Omega}^n} \sum_{\tilde{f} \in \widetilde{F}} \delta^{\omega;\tilde{f}} \cdot \Delta^n_{\omega;\tilde{f}}.
\end{aligned} \tag{75}$$

$$\text{Subject to} \sum_{\tilde{f} \in \widetilde{F}} \sum_{\tilde{\pi} \in \widetilde{\Pi}^{\mathrm{d}}_{\tilde{f}}} x^{\mathrm{d}}_{\pi,\tilde{\pi}} = 1;\ \pi \in \hat{\Pi}^{\mathrm{d}}_f,\ f \in \hat{F},\ \mathrm{d} \in \mathrm{D}$$

$$\sum_{\tilde{f} \in \widetilde{F}} x_{f\tilde{f}} = 1;\ f \in \hat{F}$$

where

$$\bar{\beta}_\pi = \sum_{\tilde{\pi}} c_\ell + \beta^{\mathrm{d}}_{\pi,\ell} - \sum_{\pi} c_\ell. \tag{76}$$

A *column* refers to the new decision variable added to the objective function, and each appropriate constraint of fog-RO-CG-M-LP. New DFACs are constructed from the resulting solution of the Pricing Problem. Each new DFAC ψ generates a new variable λ_ψ, and an associated column.

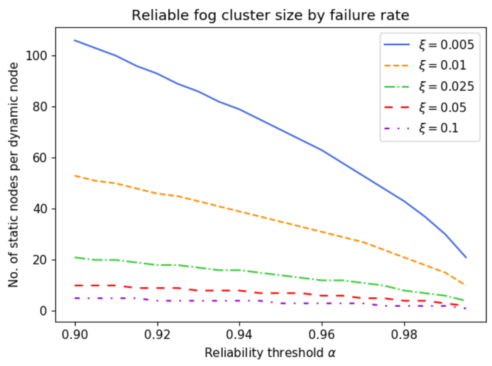


Figure 4: Maximum fog cluster size per dynamic fog node.

C. Fog-RO-CG Steps

A DFAC $\psi \in \Psi$ is composed of one fog node association, and one or more fog path associations. Therefore, the same fog associations may be present across multiple DFACs, leading to redundancies during solving. By (21), (23) and (57), we revert a DFAC ψ into fog association variables.

$$\begin{aligned}
&\psi = \left(\tilde{\mathscr{f}}(f), \widetilde{\boldsymbol{\varpi}}^{\mathrm{IN}}_f, \widetilde{\boldsymbol{\varpi}}^{\mathrm{OUT}}_f \right) \\
&\implies \left(x_{f,\tilde{f}}, \bigcup_{\pi \in \hat{\Pi}^{\mathrm{IN}}_f} x^{\mathrm{IN}}_{\pi,\tilde{\pi}}, \bigcup_{\pi \in \hat{\Pi}^{\mathrm{OUT}}_f} x^{\mathrm{OUT}}_{\pi,\tilde{\pi}} \right) = \mathbb{1}.
\end{aligned} \tag{77}$$

From Ψ', we obtain a set of candidate fog association decision variables. Finally, fog-RO-MILP is solved with candidate fog association decision variables only. Therefore, the solution of fog-RO-CG is fog-RO-MILP restricted with variables obtained from column generation. The fog-RO-CG process is summarized.

1) Initialize fog-RO-CG-M with a feasible set $\Psi' = \Psi_0$.
2) Solve fog-RO-CG-M with Ψ'.
3) Construct and solve fog-RO-CG-P from the dual variables in 2). Assemble possible DFACs from the solution, and evaluate their reduced cost (69).
4) If the reduced cost is negative, construct a new DFAC ψ, a variable λ_ψ and an associated column. Add $\Psi' = \Psi' \cup \psi$.
5) If at least one new DFAC is constructed, return to 2). Else, continue.
6) Convert Ψ' into candidate fog association variables, and solve the restricted fog-RO-MILP.

VIII. Simulation & Analysis

We are most interested in investigating the tradeoff between time efficiency and design & dimensioning cost of our selected algorithms. As a benchmark, we introduce a greedy algorithm approach to design & dimensioning, called *fog-RO-Greedy*. In order to avoid a large fixed CAPEX cost, we aim to minimize the number of dynamic fog nodes to include in the design. Our approach is to select a dynamic fog node, and associate as many static fog nodes to it while satisfying the failure probability threshold.

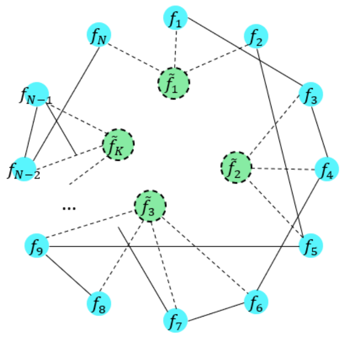


Figure 5: Small-world network topology for simulation.

Table V: Expected cost and maximum serviceable traffic load of each modeled infrastructure.

Infrastructure	**Expected Cost**	**Max Load Probability**
Static ($\rho = 95$)	437.64	0.486
Robust ($\rho = 80$)	443.25	0.999
Robust ($\rho = 60$)	439.77	0.999

A. Simulation Setup

Fig. 4 shows the maximal number of static fog nodes each dynamic fog node can service for varying failure rates and reliability thresholds. For simulated failure/reliability parameters $\xi = 0.05$ and $\alpha = 0.95$, a single dynamic fog node can support up to 7 static fog nodes. Therefore, we generate $\lceil|\hat{F}|/3.5\rceil$ dynamic fog nodes, each with at least two dynamic fog links to ensure more than sufficient candidate dynamic fog nodes.

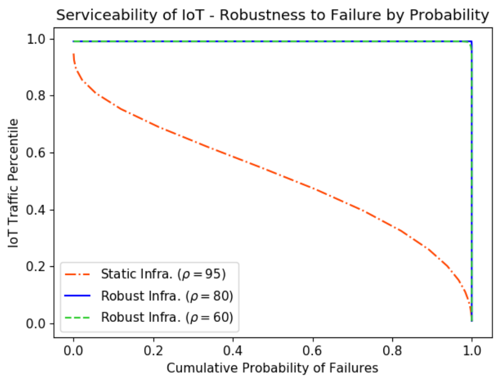
We construct a test infrastructure of static fog nodes and candidate dynamic fog nodes to evaluate the efficiency of our algorithms. The static fog nodes are first arranged in a small-world topology [26], [27]. The generated dynamic fog nodes are placed throughout the network, and connected to nearby static fog nodes via a candidate fog link. The resulting structure is shown in Fig. 5.

The simulation parameters were selected to approximate the optimal design parameters from our previous static fog network design [8]. We assume on-demand nodes consume energy at a significantly reduced rate while dormant [28], and assign an active and dormant operation cost to each node proportional to this energy consumption. The simulated results were evaluated on a computer with 24GB of RAM, and a i5-7500 CPU at 3.4 GHz.

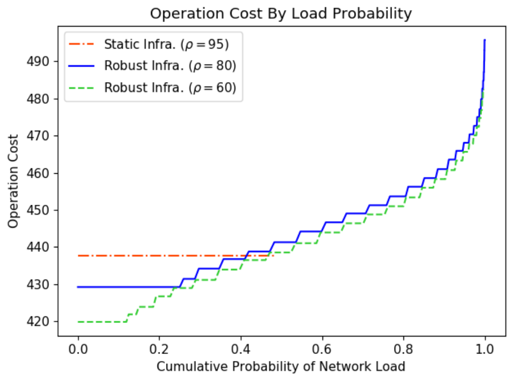
B. Validation of Robust Infrastructure

We validate our reliability approach by comparing the efficiency of static and robust infrastructures during operation. Each infrastructure has static nodes designed to support ρ-percentile of estimated IoT traffic. The static infrastructure is designed with $\rho = 95$, and has no dynamic fog nodes. We consider two robust infrastructures with $\rho \in \{60, 80\}$, and and minimal dynamic nodes to support node failures by (38).

Fog node failures are independent of each other with maximum failure rate ξ. Therefore, the total number of failures over a fog infrastructure can be approximated by a binomial distribution $\mathcal{B}(|\hat{F}|, \xi)$.



(a)



(b)

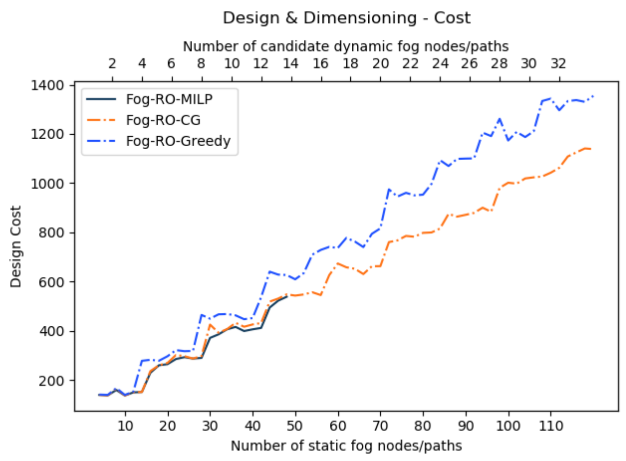
Figure 6: For failure rate $\xi = 0.05$, we compare the robustness of a static infrastructure and two dynamic infrastructures with reliability threshold $\alpha = 0.95$. We consider a) the maximum serviceable IoT traffic by failure probability, and b) the operation cost by network load probability.

Fig. 6a shows the maximum amount of estimated IoT traffic each infrastructure can support for a cumulative probability of failures. The relationship between serviceable IoT traffic percentile and cumulative probability of failures for the static infrastructure is approximately linear. Therefore, a static infrastructure is not well-equipped to operate under an expected number of fog node failures. However, by (38) with $\alpha = 0.95$, the implemented number of dynamic nodes for the static infrastructure ensures high serviceability of IoT traffic under the most probable failure situations.

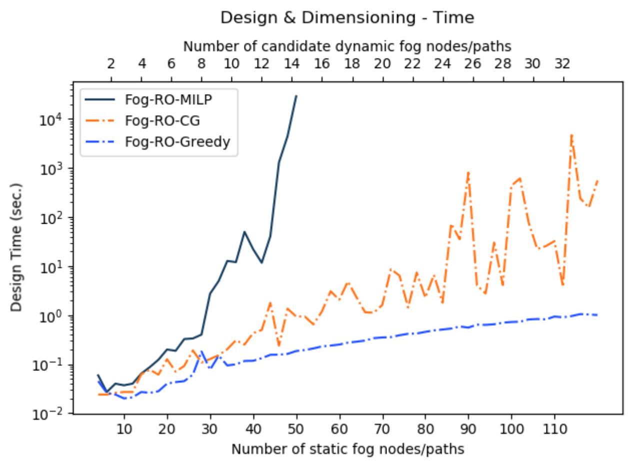
The *network load* $\mathcal{L}$ is the ratio between IoT resource requirements and static node capacity, and is defined by

$$\mathcal{L} = \max_{r \in R} \left\{ \frac{\sum_{t \in T} p_t^r}{\sum_{f \in \hat{F}_A} z_f^r} \right\} \tag{78}$$

where $\hat{F}_A$ is the set of static nodes $f \in \hat{F}$ that remain active, i.e., have not failed. If the network load surpasses 100%, then one or more dynamic fog nodes are required to support the overloaded system. The probability distribution of network loads considers different combinations of IoT traffic and fog node failures that result in the same network load.



(a) Solution cost



(b) Solution time

Figure 7: Comparison of the solution cost and time (log-scale) of each algorithm over increasing network size. Fog-RO-MILP exhibits the lowest design cost, but has exponential growth in design time. Fog-RO-CG has near optimal design cost and tractable design time.

Fig. 6b shows the operation cost of each infrastructure over the cumulative probability of network load. The operation cost of the static infrastructure remains constant since all nodes remain active. However, many node failures and high IoT traffic cause the network load to increase beyond what the static infrastructure can support with under 0.5 probability. The robust infrastructures can support the full network load with 0.999 probability. We present the expected cost, and the maximum probability of serviceable network load in Table V.

C. *Comparison of solution efficiency*

The efficiency of fog-RO-MILP and fog-RO-CG are evaluated over the solution cost and solution time. Fog-RO-Greedy is used as a benchmark in our comparisons.

1) Solution Cost: Fig. 7a shows the evaluated results of the solution cost of our algorithms. Fog-RO-Greedy provides an upper bound for the solution cost. Comparatively, fog-RO-CG closely approximates the exact cost of fog-RO-MILP. On average, the cost of fog-RO-CG is within 2.66% of fog-RO-MILP, with a minimum difference of 0.0% and a maximum difference of 4.21%. Therefore, fog-RO-CG is quite sensitive to the original problem configuration.

By Fig. 4, a higher reliability threshold α requires more dynamic fog nodes, hence higher robustness and higher design & dimensioning costs. Therefore, there is a clear tradeoff between network reliability and design & dimensioning cost.

2) Solution Time: Fig. 7b shows the evaluated results of the solution time of our algorithms. For small fog network sizes, fog-RO-MILP terminates in comparative time to fog-RO-CG. However, as the size of the network increases, the solution time of fog-RO-MILP increases drastically to a point where it cannot be feasibly computed. In contrast to this exponential growth of fog-RO-MILP, the solution time of fog-RO-CG remains polynomial as a function of network size.

Similar to the solution cost, the solution time of fog-RO-CG is heavily impacted by the original problem configuration. For large networks, fog-RO-CG is undeniably quicker than fog-RO-MILP, though the extent of the time decrease varies.

3) Remarks: Current contributions that use on-demand fog nodes do not have rerouting schedules planned for fog data distribution [20], [21]. Hong et al. [21] activate dynamic fog nodes in the vicinity of an overloaded or failed fog node. Though the latency is minimized, the dynamic allocation cost is analogous to that of fog-RO-Greedy. Zhou et al. [20] incentivize nearby vehicles to act as on-demand fog nodes through a contract negotiation process. Though this approach may results in higher profit from vehicles, it may also result in higher costs for IoT users. Furthermore, the availability of on-demand fog nodes is dependent of a supply of vehicles willing to participate. On the other hand, fog-RO-CG provides a scalable, low-cost and low-latency configuration for fog node and path association to reliable and guaranteed dynamic support.

Applying column generation to the set partitioning problem succeeds in drastically reducing the solution time while maintaining a reasonably tight approximation to the exact solution cost. However, as seen in Fig. 7, both the solution cost and time can vary based on the ratio of static to dynamic fog nodes and paths. Furthermore, generating columns with DFACs that combine node and path configurations may restrict the discovery of node and path configurations if generated separately. One alternative is to formulate a column generation model with several subproblems for fog node associations, fog path associations, and design & dimensioning configurations. Another alternative is to formulate the problem with Benders decomposition which is validated as an efficient approach to other combinatorial optimization problems [29]. Exploring these methods for this set partitioning problem is left for future work.

IX. Conclusion

In this article, we modeled the effects of IoT traffic and static fog configurations on the network congestion and latency. We used this model to propose the fog-RO-MILP model for the cost-efficient design & dimensioning of a reliable and fault-tolerant fog infrastructure. To overcome the intractability of fog-RO-MILP, we proposed fog-RO-CG, a scalable column generation method to approximate the minimal implementation cost of fog-RO-MILP.

We validated the operation efficiency of the proposed infrastructure with our previous static fog infrastructure [8]. The operation cost and IoT traffic serviceability of the proposed infrastructure remains stable over a high cumulative probability of network load and node failures. We compared simulated results of our methods, alongside a benchmark greedy approach (fog-RO-Greedy), and observed fog-RO-CG closely approximates the optimal cost of fog-RO-MILP, while executing in polynomial time. Compared to other contributions that use on-demand fog nodes, our solution provides optimal allocation costs with guaranteed reliability.

In future work, we will continue our expansion of the scope of fog design & dimensioning to include fog node mobility. For a fixed fog infrastructure comprised of both static and dynamic fog nodes, we will model the addition of mobile fog nodes to existing infrastructure. We will study the cost sensitivity of a fog design to additional fog nodes that geographically extend the IoT reach of the fog infrastructure. Our current methodologies are based on theoretical estimates of IoT traffic patterns; we intend to further simulate our method on a concrete set of historic IoT traffic data to gain more accurate insight, and better simulations of network conditions. Finally, we intend to build a small-scale practical implementation of a reliable fog infrastructure. These additional features on our fault-tolerant fog design & dimensioning methodology will further encourage the adoption and implementation of wide-scale fog systems.

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [PGSD3 - 558695 - 2021].

References

[1] IDC, "The growth in connected iot devices is expected to generate 79.4zb of data in 2025, according to a new idc forecast," 2019. Accessed: 2022-03.

[2] C. S. Inc., "Fog computing and the internet of things: Extend the cloud to where the things are [white paper]," 2015.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[4] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn, *et al.*, "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures," *Journal of Cloud Computing*, vol. 7, no. 1, p. 16, 2018.

[5] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149–160, 2019.

[6] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.

[7] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management and evaluation of fog computing systems: A survey," *IEEE Internet of Things Journal*, 2020.

[8] I. Martinez, A. Jarray, and A. S. Hafid, "Scalable design and dimensioning of fog-computing infrastructure to support latency sensitive iot applications," *IEEE Internet of Things Journal*, 2020.

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

[10] R. Marino, C. Wisultschew, A. Otero, J. M. Lanza-Gutierrez, J. Portilla, and E. de la Torre, "A machine-learning-based distributed system for fault diagnosis with scalable detection quality in industrial iot," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4339–4352, 2020.

[11] B. Natesha and R. M. R. Guddeti, "Fog-based intelligent machine malfunction monitoring system for industry 4.0," *IEEE Transactions on Industrial Informatics*, 2021.

[12] U. Ozeer, X. Etchevers, L. Letondeur, F.-G. Ottogalli, G. Salaün, and J.-M. Vincent, "Resilience of stateful iot applications in a dynamic fog environment," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pp. 332–341, 2018.

[13] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, 2019.

[14] E. Balas and M. W. Padberg, "Set partitioning: A survey," *SIAM review*, vol. 18, no. 4, pp. 710–760, 1976.

[15] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[16] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.

[17] B. Do Chung, T. Yao, C. Xie, and A. Thorsen, "Robust optimization model for a dynamic network design problem under demand uncertainty," *Networks and Spatial Economics*, vol. 11, no. 2, pp. 371–389, 2011.

[18] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," *Mathematics of operations research*, vol. 23, no. 4, pp. 769–805, 1998.

[19] B. S. Pimentel, G. R. Mateus, and F. A. Almeida, "Stochastic capacity planning and dynamic network design," *International Journal of Production Economics*, vol. 145, no. 1, pp. 139–149, 2013.

[20] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.

[21] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, pp. 15–20, 2013.

[22] C. Shang, F. Yang, X. Gao, X. Huang, J. A. Suykens, and D. Huang, "Concurrent monitoring of operating condition deviations and process dynamics anomalies with slow feature analysis," *AIChE Journal*, vol. 61, no. 11, pp. 3666–3682, 2015.

[23] A. J. Jara, D. Genoud, and Y. Bocchi, "Big data in smart cities: from poisson to human dynamics," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 785–790, IEEE, 2014.

[24] T. Hoßfeld, F. Metzger, and P. E. Heegaard, "Traffic modeling for aggregated periodic iot data," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 1–8, IEEE, 2018.

[25] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, "Fundamentals of queueing theory," ch. 6.1, pp. 255–259, John Wiley & Sons, 2018.

[26] Z. Wang, A. Scaglione, and R. J. Thomas, "Generating statistically correct random topologies for testing smart grid communication and control networks," *IEEE transactions on Smart Grid*, vol. 1, no. 1, pp. 28–39, 2010.

[27] A. Barrat and M. Weigt, "On the properties of small-world network models," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 13, no. 3, pp. 547–560, 2000.

[28] O. Onireti, A. Mohamed, H. Pervaiz, and M. Imran, "Analytical approach to base station sleep mode power consumption and sleep depth," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, IEEE, 2017.

[29] A. M. Costa, "A survey on benders decomposition applied to fixed-charge network design problems," *Computers & operations research*, vol. 32, no. 6, pp. 1429–1450, 2005.

Chapter 5

Design & Dimensioning of a UAV Set Covering in High-Traffic IoT-Fog Environments

Abstract The fog-computing paradigm provides low-latency processing and storage between Internet of Things (IoT) applications and Cloud data centres. Normal IoT activity may produce infrequent yet substantial spikes in user traffic, which would require a large static fog infrastructure to service. Instead, we consider the viability of having a smaller static fog infrastructure, and supplementing additional traffic spikes with fog-enabled unmanned aerial vehicles (fog-UAVs). This article formulates the optimal design & dimensioning of fog-UAVs and fog-UAV charging/deployment stations as a probabilistic location set covering problem (PLSCP). We model the fog-UAV PLSCP as a mixed-integer linear program (MILP), from which we derive several relaxed models including one based on the Benders decomposition technique. Finally, we simulate our models over a set of city-wide IoT hotspots with various traffic percentile thresholds, and evaluate our results.

This chapter extends previous chapters by considering ‘bursty’ IoT traffic, and using fog-UAVs to support an existing fog infrastructure. This chapter has been submitted to IEEE Internet of Things [41].

Design & Dimensioning of a UAV Set Covering in High-Traffic IoT-Fog Environments

Ismael Martinez, Abdelhakim Senhaji Hafid, Michel Gendreau

***Abstract*—The fog-computing paradigm provides low-latency processing and storage between Internet of Things (IoT) applications and Cloud data centres. Normal IoT activity may produce infrequent yet substantial spikes in user traffic, which would require a large static fog infrastructure to service. Instead, we consider the viability of having a smaller static fog infrastructure, and supplementing additional traffic spikes with fog-enabled unmanned aerial vehicles (fog-UAVs). This article formulates the optimal design & dimensioning of fog-UAVs and fog-UAV charging/deployment stations as a probabilistic location set covering problem (PLSCP). We model the fog-UAV PLSCP as a mixed-integer linear program (MILP), from which we derive several relaxed models including one based on the Benders decomposition technique. Finally, we simulate our models over a set of city-wide IoT hotspots with various traffic percentile thresholds, and evaluate our results.**

***Index terms*—** Fog computing, fog design & dimensioning, Internet of Things (IoT), mixed-integer linear programming (MILP), Benders decomposition, probabilistic set covering, unmanned aerial vehicles.

I. Introduction

NETWORKS of collaborative Internet of Things (IoT) devices are expected to be at the forefront of next-generation Smart technology [1]. The data from these networks, composed of several sensors and actuators, are often processed by Cloud data centres. However, network congestion and large distance between IoT and Cloud may result in high response time, which is not ideal for latency sensitive IoT applications such as in health care [2], autonomous vehicles [3], and multimedia [4]. As a result, the implementation of fog infrastructures around latency sensitive IoT has become a recent focus of research [5].

Individual fog nodes are characterized as micro-data centres with predominantly wireless access that provide compute, storage and networking services to the IoT environment. The fog network is a highly virtualized platform of heterogeneous and geographically distributed fog nodes. Wide-spread geo-distribution creates a network of fog nodes on the network edge that can communicate and interact with IoT in real-time. Therefore, expanding the fog network is an effective way to efficiently support IoT requests with minimal latency [6].

I. Martinez is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ismael.martinez@umontreal.ca).

A. S. Hafid is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ahafid@iro.umontreal.ca).

M. Gendreau is with the Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Quebec, H3C 3A7, Canada (email:michel.gendreau@polymtl.ca)

The planned construction of a new fog infrastructure involves a) *design* — the location of fog nodes, and b) *dimensioning* — the quantity of resources per fog node. Our previous work has focused on the design & dimensioning of a static fog network to service nearby IoT requests [7]. With the possibility that fog nodes may shut down due to failure or maintenance, further on-demand nodes were included in the fog design to create a robust fog network [8].

When IoT traffic is dictated by fixed request schedules by IoT devices, then IoT traffic is predictable and consistent throughout the day. However, if IoT requests are governed by human behavior, it can result in periods of volatile and abnormally high IoT traffic [9]. In this case, an infrastructure composed of only geographically static fog nodes is unfit to support the high variability of IoT traffic [8], [9].

We extend our previous work on fog design & dimensioning [7], [8] to consider *dynamic environments*, which include 1) mobile fog devices, and 2) large fluctuations in IoT traffic, i.e., 'bursty' traffic. Specifically, we consider fog-enabled unmanned aerial vehicles, known as *fog-UAVs*, to move to areas of unexpectedly high service demand.

In this paper, fog-UAVs are deployed to areas of high IoT traffic to support the overloaded static fog infrastructure. The design & dimensioning of fog-UAVs is modeled as a probabilistic location set covering problem (PLSCP) [10], [11] where we find a minimal set of fog-UAVs to support any overloaded traffic area with high probability. This problem is first modeled as a mixed-integer linear program (MILP) from which multiple relaxed models are derived, including one based on Benders decomposition. These models are compared alongside a benchmark for scalability and cost.

Our contributions include

- A geographic estimation of 'bursty' IoT traffic.
- An exact, a relaxed, and a Benders decomposition-based MILP of design & dimensioning for the fog-UAV PLSCP.
- A simulated analysis of IoT traffic serviceability and solution performance of our models over varying overload probability thresholds.

The remainder of this article is organized as follows. Section II reviews related literature. Section III describes the problem statement, and estimates the IoT traffic and busy fractions required for the fog-UAV PLSCP. Section IV describes the fog-UAV MILP exact and relaxed models. Section V formulates a Benders decomposition of the relaxed model, and a heuristic model as a simulation benchmark. Section VI executes all models over increasing number of hotspots and stations, and evaluates the solution time and cost results. Finally, section VII discusses future work, limitations and concludes the paper.

II. Related Work

A. Design & Dimensioning

Martinez et al. [7] propose a design & dimensioning model for the general IoT landscape. The resulting fog infrastructure is equipped to support all resource requirements of IoT traffic within a latency-threshold. However, a) a static estimation of IoT traffic is used, and b) all fog nodes are assumed to be reliable. These limitations are addressed in [8] where on-demand fog nodes are used to either replace and restore data from failed fog nodes, or provide additional support to an overloaded system. A sensitivity analysis over the number of failures and IoT load of the network shows that the addition of on-demand fog nodes can guarantee reliable service to IoT even in the worst cases.

In both [7], [8], an exact yet intractable MILP formulation is compared to a heuristic column generation (CG) model. The CG models approximate the MILP well, though the tightness of this approximation can vary wildly based on the initial parameter configuration. Therefore, we formulate the PLSCP using Benders decomposition for its constraint generating properties [12]. Furthermore, the on-demand fog nodes are statically located [8], which restricts their overload support by location. Therefore, in this paper, we consider the use of fog-UAVs to increase flexibility of overload support to a larger physical region.

B. Probabilistic Location Set Covering Problem

Chapman and White [13] introduced the PLSCP for reliable service coverage. This problem is now considered one of the fundamental model of integer programming [14]. Often applied to emergency facilities such as police, fire, and ambulance stations [15], [16], [17], this problem seeks to place minimal facilities around a city such that every user has at least one nearby facility that can provide emergency services. This problem represents service reliability through chance constraints derived from *busy fractions*, i.e., the probability that an emergency vehicle is unavailable. Though Chapman and White assume a uniform demand distribution, and thus uniform busy fractions, Revelle and Hogan [11] partition the city into sectors with different busy fractions. By doing so, they achieve a more flexible and accurate estimation of the provided reliability of a set covering design.

In this paper, we model the design & dimensioning of fog-UAVs as a PLSCP, where fog-UAVs are positioned to support infrequent and geographically sparse bursts of IoT traffic. We use the mobility and service radiuses of fog-UAVs to define reachable sectors around charging stations, and likewise use expected IoT demand around each charging stations to define the fog-UAV busy fractions.

C. Unmanned Aerial Vehicles

The mobility of unmanned aerial vehicles (UAVs) has been shown to be useful in IoT environments. Recent use cases use UAVs i) to support overloaded road-side units (RSUs) in Internet of Vehicle (IoV) environments [18], ii) as a Wi-Fi chain to provide internet to users in disaster areas [19], and iii) to support energy harvesting and computation offloading of mobile users in a mobile edge-computing (MEC) environment [20]. These applications take advantage of the swift, flexible and on-demand deployment of UAVs as aerial communication servers. However, UAVs do not have a reliable power supply and need to recharge often [21]. Therefore, the placement of UAV charging stations must be considered when using UAVs as computational support.

The optimal hovering locations of UAVs for nearby user support has previously been formulated as a set covering problem [22], [23], [24]. Chiaraviglio et al. [22] deploy a 5G network over a rural area using UAVs and ground charging stations. To provide continuous service to each area, UAVs alternate between hovering and recharging, and must be in communication range of ground stations for additional connectivity. This ensures all areas are provided 5G network access at all times. However, using UAVs in continuous rotation over a locked region does not make the best use of the flexibility of UAV technology, and may be costly.

Park et al. [23] focus only on the hovering locations of UAVs over an emergency network of users, and do not consider the charging/deployment stations of UAVs. UAVs are not tethered to any station, and can hover in any 2-dimensional coordinate relative to demand points. However, not considering from where UAVs are deployed or for how long a UAV can hover on a single charge limits the applicability of this approach.

Mao et al. [24] designs a UAV-based emergency medical service network that optimizes the mean waiting time of demands due to UAV response and deployment. Waiting times are simulated from demands with Poisson arrival and serviced through a queue of available UAVs deployed from charging stations. The medical services are physically provided on site by a UAV, limiting this application to one incident per UAV at a time. Similar to [23], all services are assumed to be feasibly provided under a single battery charge.

We compare these three UAV set covering applications [22], [23], [24] with our fog-UAV contribution in terms of a) deployment flexiblity, b) available UAV battery charging, and c) untethered UAV mobility. In all three cases, all demand spots are covered with a minimal set of UAVs [22], [23], [24]. For our purposes, we assume that IoT traffic may be bursty, and unpredictable in duration and location.

1) Deployment flexibility: Unlike [22], [23] who cover all demand points at once for an extended period, we minimize the design cost of a fog-UAV infrastructure by allowing scarce and flexible deployment of fog-UAVs to multiple regions.

2) UAV battery charging: Similar to [22], we account for the battery consumption of UAVs and provide charging stations to ensure no hovering UAV fails due to a depleted battery. Multiple fog-UAVs are available at each station so that any hovering fog-UAV with low battery may be replaced.

3) Untethered UAV mobility: Unlike [23] who requires UAVs to remain a communication distance from ground stations, we allow UAVs to move freely and untethered from charging stations, constrained only by their own battery capacity [23], [24]. The summary of these comparisons is shown in Table I.

Table I: Comparison of fog-UAVs with other UAV set covering solutions.

Consideration/Feature	**[22]**	**[23]**	**[24]**	**Fog-UAV**
Deployment flexibility	✗	✗	✓	✓
UAV battery recharging	✓	✗	✗	✓
Untethered UAV mobility	✗	✓	✓	✓

III. Mobile fog design

In this section, we provide a detailed description of our problem, including a city-wide estimation of the IoT traffic, and a compact expression to describe when a static fog infrastructure is overloaded.

A. Problem Statement

Based on previous work [7], [8], we suppose that a physical region is supported by a fog infrastructure composed of statically placed nodes. This fog infrastructure is equipped to service mid to high levels of IoT requests by IoT users. It is possible for IoT traffic in any given area to momentarily peak beyond the serviceable limits of the static fog infrastructure. In this scenario, the region is said to be *overloaded*, and requires additional support.

We propose to use fog-UAVs to service overloaded areas. Until they are required, each fog-UAV is based in a nearby station where it charges its battery. When needed, the fog-UAV is deployed from the station, and may return either to the original station or to a nearby station once service is complete. A fog-UAV may need to return to a station before the IoT service is complete, such as for charging or maintenance. In this case, our design ensures that there are additional fog-UAVs ready for deployment to complete unfinished service. All IoT devices within the fog-UAV service radius have direct one-hop communication with the fog-UAV, thus minimizing service latency. The validity of one-hop communication for varying fog-UAV hovering height and environmental interference is left for future work.

We assume all stations have both charging and deployment capabilities. For a set region of IoT users, we simultaneously a) select the locations for stations, and b) select a set of UAVs to service the area, each assigned to a subset of total stations for charging and deployment.

B. IoT Traffic Estimation

Geographic and demographic factors of a region can provide a strong indication of associated IoT traffic. Within a city, it is reasonable to estimate the IoT traffic from any given borough to be proportional to its population [25]. Intuitively, such an assumption would estimate higher IoT traffic from highly populated areas, and therefore emphasize these dense areas in a set cover. Jara et al. [9] consider the effects of Human Dynamic within a Smart City on the traffic level of IoT, and conclude that even in the presence of 'bursty' data in complex IoT networks, IoT traffic can still be well approximated by a Poisson distribution.

We use these concepts of 1) proportionality of traffic with population, and 2) Poisson traffic over a geographic region, to partition a geographic region into clusters, known as *hotspots*, of K people with approximately equal population and IoT traffic. Note, a larger cluster size K results in fewer hotspots, each with higher resource demand, which is more manageable for the fog-UAV PLSCP. However, since hotspots are disjoint aggregations of the population, larger hotspots may remove valuable variability in population density within a hotspot or between hotspots.

Using the city of Montreal as an example with $K = 10,000$, we gathered the populations of each of Montreal's boroughs [26], rounded to the nearest integer multiple of K. A borough with a rounded population of nK is partitioned into n hotspots of size K. These hotspots are uniformly placed throughout the borough. The resulting hotspot map is shown in Fig. 1a. The aggregated set of all hotspots from all boroughs is denoted H.

Let H be the set of hotspots. For simplicity, we assume the IoT traffic by population is consistent throughout the city. Thus, the IoT traffic throughout the city can be modeled as a set of independent and identically distributed (i.i.d.) $Poisson(\lambda)$ distributions with traffic arrival rate λ, one associated to each hotspot. The IoT traffic aggregated over H also follows a Poisson distribution such that

$$\begin{aligned} & T_h \sim Poisson(\lambda), \quad h \in H \\ \implies & \sum_{h \in H} T_h \sim Poisson(|H| \cdot \lambda), \end{aligned} \tag{1}$$

where T_h represents the number of requests from hotspot h.

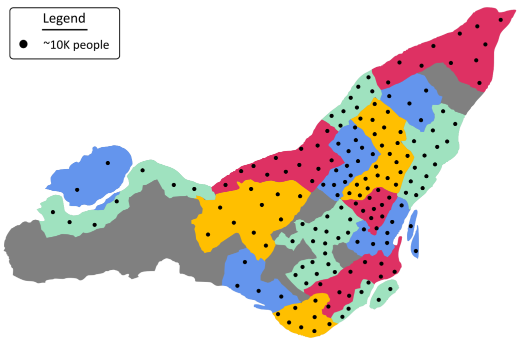
We consider the dimensioning of CPU, RAM, and storage resources (STR) within each fog-UAV. For resources $R = \{\text{CPU}, \text{RAM}, \text{STR}\}$, each hotspot $h \in H$ has request resource requirement mean $\overline{\psi}_h^r$, $r \in R$. If each fog-UAV can service a limited number of hotspots, then more populated regions would require more fog-UAV covers. Fig. 1b gives an example of a fog-UAV cover over a densely populated region.

C. Busy fraction

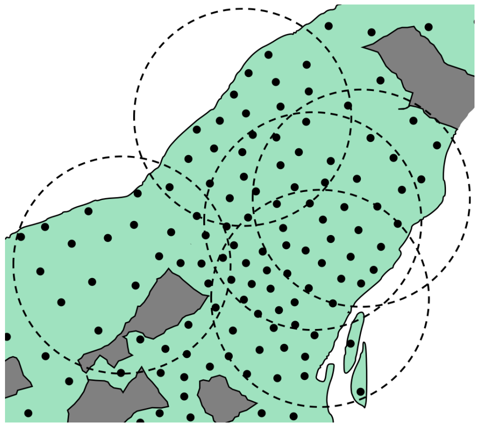
Chapman and White [13] use the concept of busy fractions to represent the fraction of time, and therefore the probability, that an emergency vehicle was unavailable to service a new demand. We use this concept to denote the probability that the static fog infrastructure is saturated and thus unavailable to service a new IoT request.

Let $\mathcal{P}(t;\ \lambda)$ be the cumulative Poisson distribution up to t requests, and let $\mathcal{P}^{-1}(p;\ \lambda)$ be the p-percentile traffic. We assume that the static fog infrastructure can service IoT traffic up to the β-percentile, $\beta \in [0.5, 1)$. We define T_β as the maximum number of requests in the β-percentile. That is,

$$T_\beta = \mathcal{P}^{-1}(\beta;\ |H| \cdot \lambda). \tag{2}$$



(a) Hotspot representation of data



(b) Example set covering

Figure 1: The population of Montreal, Canada, subdivided into boroughs. IoT and mobile traffic demand is proportionally estimated by geographic and demographic factors [25], [26].

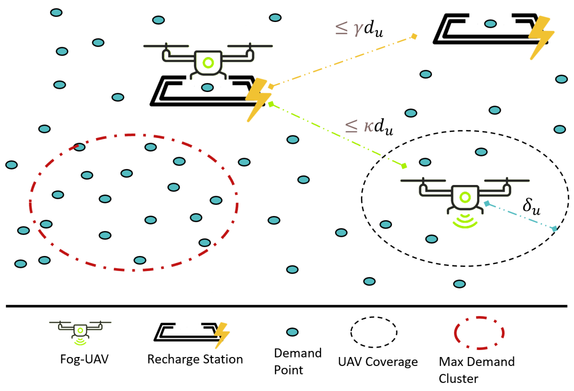


Figure 2: Overview of the fog-UAV architecture.

Consider a set H' of adjacent i.i.d. hotspots, $H' \subset H$. Then,

$$\begin{aligned} &P(\sum_{h\in H} T_h \leq T_\beta) = \beta \\ \implies &P\left(\frac{|H'|}{|H|}\sum_{h\in H} T_h \leq \frac{|H'|}{|H|}T_\beta\right) = \beta \\ \implies &P\left(\sum_{h\in H'} T_h \leq \frac{|H'|}{|H|}T_\beta\right) = \beta \end{aligned} \tag{3}$$

Hence, the probability that the static fog infrastructure can service a subset $H' \subseteq H$ of adjacent hotspots is also β. We say a subset H' is *overloaded* if its IoT traffic surpasses the capacity of the static fog infrastructure, i.e., β-percentile. The probability that H' becomes overloaded is therefore $1-\beta$.

When a fog-UAV is deployed to assist an overloaded set of hotspots, the fog-UAV becomes unavailable to other hotspots. Therefore, $1-\beta$ is also the busy fraction of fog-UAVs, i.e., the probability that a fog-UAV is unavailable.

IV. Fog-UAV-MILP Model

Let S be the set of all candidate station locations, and let U be the set of all candidate fog-UAVs. We define the following binary decision variables:

$$x_s = \mathbb{1}[\text{Station } s \in S \text{ is optimal}] \tag{4}$$
$$y_u = \mathbb{1}[\text{UAV } u \in U \text{ is optimal}] \tag{5}$$
$$p_h^u = \mathbb{1}[\text{Hotspot } h \in H \text{ is serviced by UAV } u] \tag{6}$$
$$w_s^u = \mathbb{1}[\text{UAV } u \text{ is deployed from station } s] \tag{7}$$

where $\mathbb{1}[\cdot]$ is the indicator function. By definition of (7) and (6), w_s^u serves a linking variable between x_s and y_u while p_h^u links each hotspot h to some y_u.

We define the continuous dimensioning variable z_u^r, $u \in U, r \in R$ such that $z_u^r \in [0, Z_u^r]$ for a predefined upper bound Z_u^r. This variable defines the quantity of resource r assigned to fog-UAV u to help support overloaded IoT traffic.

We acknowledge the battery restrictions of a UAV would not allow the UAV to fly indefinitely. By [27], electric vehicle charge linearly over the first ~80% of its battery charge. For simplicity, we assume a UAV on a full battery has a fixed flight time, which does not degrade with multiple charges, and depletes linearly during its flight. Hence, for simplicity of mobile fog design, we assume at least one UAV per station is at 80% charge and ready for deployment at any given time.

We allow the locations of UAVs to be flexible, as long as they stay within a distance proportional to their battery capacity from a charging station. We define $\eta_u > 0$ as the distance a UAV can travel on a full battery, and $\delta_u > 0$ as the communication radius of u. We assume any IoT hotspot within the communication radius of a hovering fog-UAV u is serviced in one-hop and minimal latency. A visual representation of stations, fog-UAVs and hotspots is presented in Fig. 2. Finally, each fog-UAV $u \in U$ is associated to a default station $s \in S$. We define this relationship as a mapping $\xi : U \to S$ where $\xi(u) \in S$ is the default station of u.

A. *Pre-processing*

Prior to composing the fog-UAV-MILP model, we need to understand the interactions between fog-UAVs, stations and hotspots. First, we calculate all the distances $d_{u,h}$ between fog-UAV $u \in U$ and hotspot $h \in H$, and the distances $d_{s,s'}$ for every pair of stations $(s, s') \in S^2$.

We are also interested in the maximum number of hotspots within each fog-UAV radius for a range of hovering locations.

Table II: Fog-UAV-MILP Parameters

Variable	Description
α	The fog-UAV service threshold.
β	The static fog coverage percentile.
η_u	The maximum flight distance $u \in U$ can travel on a single battery.
δ_u	The communication radius of $u \in U$.
$\xi(u)$	The associated default station of $u \in U$.

Table III: Fog-UAV-MILP Precomputed Parameters

Variable	Description
$d_{\xi(u),s}$	The distance between a station $s \in S$ and a default station $\xi(u)$.
d_h^s	The distance between a station s and a hotspot h.
H_u^c	Set of hotspots in the service region of $u \in U$ centred at $c = (a, b)$.
$\mathcal{C}_u^h$	Set of hotspots that can be serviced simultaneously with $h \in H$ by $u \in U$.
$\mathcal{V}$	The set of viable UAV-station-hotspot triples.

Suppose each hotspot h is positioned at a relative coordinate $c_h = (a_h, b_h) \in \mathbb{R}^2$. For a relative position $c = (a, b) \in \mathbb{R}^2$ of UAV $u \in U$, we define the set of hotspots it can service as

$$H_u^c = \{h \in H \mid d_h^c \leq \delta_u\} \tag{8}$$

where d_h^c is the distance between coordinate $c = (a, b)$ and h. The maximum request arrival rate Λ_h^u and requirement $\Psi_{u,h}^r$ per resource $r \in R$ from hotspos within some δ_u-radius cover containing h are

$$\Lambda_h^u = \max_{c\in\mathbb{R}^2} \left\{ |H_u^c|\lambda \mid h \in H_u^c \right\} \tag{9}$$

$$\Psi_{u,h}^r = \max_{c\in\mathbb{R}^2} \Big\{ \sum_{h'\in H_u^c} \overline{\psi}_{h'}^r \mid h \in H_u^c \Big\}; \quad r \in R. \tag{10}$$

Table III summarizes the variables and sets that are precomputed prior to building the fog-UAV-MILP model.

B. Objective Function

The fog-UAV-MILP can be described as *the design & dimensioning cost of a fog-UAV system, given every hotspot is serviceable with high probability.*

We define positive scalars C_s and V_u as the capital expenditure costs of station $s \in S$ and fog-UAV $u \in U$ respectively, and v_u^r is the unit resource dimensioning cost of $r \in R$ in u. The objective function of fog-UAV minimizes the fixed and unit costs of design & dimensioning such that

$$\text{Min} \quad \sum_{s\in S} x_s \cdot C_s + \sum_{u\in U} \big(y_u \cdot V_u + \sum_{r\in R} z_u^r \cdot v_u^r \big). \tag{11}$$

C. Mobility Constraints

Since a fog-UAV u must be deployed by its default station $\xi(u) \in S$, we are interested in finding stations from $\xi(u)$ that are reachable by u within a single battery charge. Recall, by assumption, there exists at least one fog-UAV with at least 80% battery charge available for deployment at any given time. We define constants $\kappa \in (0, 0.4]$ and $\gamma \in [0.5, 0.8)$ such that each fog-UAV u is able to move freely as long as it stays within $\kappa\eta_u$ distance from at least one associated station, and all associated stations are at most $\gamma\eta_u$ distance apart. That is,

$$d_{\xi(u),s} \cdot w_s^u \leq \gamma\eta_u; \quad u \in U, s, \in S \tag{12}$$

$$d_h^s \cdot w_s^u p_h^u \leq \kappa\eta_u + \delta_u; \quad u \in U, s \in S, h \in H. \tag{13}$$

1) Viability preprocessing: Fog-UAV-MILP resolution can be simplified by precomputing the variables that satisfy the constraint (12) and non-linear constraint (13). Hence, for $u \in U, s \in S$ and $h \in H$, we define

$$\mathcal{W} = \{(u, s) \mid d_{\xi(u),s} \cdot w_s^u \leq \gamma\eta_u\} \tag{14}$$

$$\mathcal{Q} = \{(u, s, h) \mid d_h^s \cdot w_s^u p_h^u \leq \kappa\eta_u + \delta_u\} \tag{15}$$

$$\mathcal{V} = \{(u, s, h) \mid (u, s) \in \mathcal{W} \text{ and } (u, s, h) \in \mathcal{Q}\}. \tag{16}$$

Set $\mathcal{W}$ satisfies constraint (12), and set $\mathcal{Q}$ satisfies constraint (13). Set $\mathcal{V}$ represents the **viable** triplets (u, s, h) which satisfy both distance constraints. We may use u, s or h as indices to further filter the sets as needed. For example, $\mathcal{V}_{u,s}$ is the set of viable hotspots for a fixed UAV-station pair (u, s). Furthermore, we define

$$\mathcal{H}_u = \{h \in H \mid \exists s : (u, h) \in \mathcal{V}_s\} \tag{17}$$

$$\mathcal{U}_h = \{u \in U \mid \exists s : (u, h) \in \mathcal{V}_s\} \tag{18}$$

where $\mathcal{H}_u$ is the set of reachable hotspots from $u \in U$, and $\mathcal{U}_h$ is the set of fog-UAVs that can reach $h \in H$.

D. Service Constraint

The set of fog-UAVs cannot cover all areas at once, hence we require a high probability that there are sufficient fog-UAVs to service all overloaded areas. Let $\alpha \in [0.5, 1)$ be the threshold probability that at least one fog-UAV is available to service an overloaded hotspot h. We define

$$\mathcal{C}_u^h = \{h' \in H \mid \exists c \in \mathbb{R}^2 : h, h' \in H_u^c\} \tag{19}$$

as the set of hotspots that can be serviced at the same time as h by u. Hence, the complement $\overline{\mathcal{C}}_u^h$ is the set of hotspots that cannot be serviced from fog-UAV u at the same time as hotspot h. Taking into account that a fog-UAV may service multiple hotspots, the busy fraction for u to h is $\mathcal{B}_{u,h}$ where

$$\mathcal{B}_{u,h} = (1 - \beta)\frac{|\overline{\mathcal{C}}_u^h|}{|\mathcal{H}_u|}. \tag{20}$$

To guarantee there are sufficient fog-UAVs to service each hotspot $h \in H$, we require

$$\begin{aligned} &P(\text{at least 1 fog-UAV is available}) \geq \alpha \\ \iff &1 - \prod_{u\in\mathcal{U}_h} (\mathcal{B}_{u,h})^{p_u^h} \geq \alpha \\ \iff &1 - \alpha \geq \prod_{u\in\mathcal{U}_h} (\mathcal{B}_{u,h})^{p_u^h} \\ \iff &\log(1 - \alpha) \geq \log\big(\prod_{u\in\mathcal{U}_h} (\mathcal{B}_{u,h})^{p_u^h} \big) \\ \iff &\log(1 - \alpha) \geq \sum_{u\in\mathcal{U}_h} p_u^h \log(\mathcal{B}_{u,h}). \end{aligned} \tag{21}$$

E. Design Constraints

We define two linking constraints

$$x_s \geq w_s^u; \quad (u,s) \in \mathcal{W} \tag{22}$$
$$y_u \geq w_s^u; \quad (u,s) \in \mathcal{W} \tag{23}$$
$$y_u \geq p_h^u; \quad u \in \mathcal{U}_h. \tag{24}$$

Constraints (22) and (24) respectively state that if UAV-station association w_s^u is part of the optimal solution, then both station s and fog-UAV u must also be part of the optimal solution. Similarly, constraint (24) states that if a UAV-hotspot association p_h^u is part of the optimal solution, then the fog-UAV u must also be part of the optimal solution.

Let $Y_u \in \mathbb{N}^{\geq 1}$ be the minimum number of stations associated to $u \in U$. Then, we define two UAV-station association constraints as

$$x_{\xi(u)} = y_u; \quad u \in U \tag{25}$$
$$\sum_{s \in \mathcal{W}_u} w_s^u \geq Y_u \cdot y_u; \quad u \in U \tag{26}$$

where constraint (25) ensures that any fog-UAV $u \in U$ in the optimal solution must be deployed by $\xi(u)$, and constraint (26) enforces the mobility of fog-UAVs to more than one station.

F. Dimensioning Constraints

Each fog-UAV has a maximum resource capacity, as well as an expected resource demand from IoT which are represented by the dimensioning constraint; this ensures that each dimensioning of fog-UAV u can support the worst-case IoT traffic demand around u. We define the dimensioning of u by the positive continuous decision variable $z_u^r \in [0, Z_u^r]$, where Z_u^r is its resource capacity, and let ρ be the estimated ρ-percentile IoT traffic for resource where $\rho \in (0,1)$. A fog-UAV u should satisfy additional IoT resource demand around hotspot h up to T_h^u requests per hour,

$$T_h^u = \mathcal{P}^{-1}(\rho;\ \Lambda_h^u), \tag{27}$$

where Λ_h^u is the maximum arrival rate for a cluster of hotspots from u (9). Hence, the dimensioning requirement of each fog-UAV becomes

$$z_u^r \geq tT_h^u \Psi_{u,h}^r \cdot p_h^u, \quad u \in U,\ r \in R,\ h \in \mathcal{H}_u. \tag{28}$$

for maximum mean request requirement $\Psi_{u,h}^r$ per request r from u (10), and fraction $t \in (0,1]$ of requests per hour u will process at once.

In summary, fog-UAV-MILP is defined by

$$\begin{aligned} &\text{Min } (11) \\ &\text{Subject to } \ (21) - (26),\ (28) \end{aligned}$$

G. MILP Relaxation

The fog-UAV-MILP model constraints show a clear dependency among the fog-UAVs, charging stations and hotspots. However, it is possible to simplify the model to have fewer variables, fewer constraints, and a linear relaxation on some integer variables, all while mainting the same optimal design & dimensioning cost. The relaxed model is known as *fog-UAV-MILP-R*, and is defined as

$$\begin{aligned} &\text{Min } (11) \\ &\text{Subject to } (21), (25), (28), \end{aligned}$$
$$\frac{1}{|\mathcal{H}_u|} \sum_{h \in \mathcal{H}_u} p_h^u \leq y_u; \quad u \in U \tag{29}$$
$$(Y_u - 1) \cdot y_u \leq \sum_{s \in \mathcal{W}_u \setminus \xi(u)} x_s; \quad u \in U \tag{30}$$

where x_s is relaxed to be linear in $[0,1]$ during solving. The UAV-station linking variables w are removed. Then, contraints (22)–(24) and (26) are replaced by constraints (29) and (30) by considering only viable sets $\mathcal{W}$ (15) and $\mathcal{H}$ (17) .

1) Complexity Improvements: In the worst case, fog-UAV-MILP has $|S|$ station design variables x, $|U|$ fog-UAV design variables y, $|U \times R|$ fog-UAV dimensioning variables z, $|U \times S|$ UAV-station linking variables w , and $|U \times H|$ UAV-hotspot linking variables p. That is,

$$|S| + |U| + |U \times R| + |U \times S| + |U \times H| \tag{31}$$

total variables. In addition, fog-UAV-MILP has $|H|$ constraints from (21), at most $|U \times S|$ from each of (22) and (23), at most $|U|$ constraints from each of (24), (25) and (26), and at most $|U \times R \times H|$ constraints from (28). That is

$$|H| + 2|U \times S| + 3|U| + |U \times R \times H| \tag{32}$$

total constraints.

Fog-UAV-MILP-R removes the $|U \times S|$ w variables, and replaces $|U \times S|$ constraints (22)–(24) and (23) with $2|U|$ constraints (29) and (30). By doing so, the total number of variables and constraints are reduced to

$$\begin{gathered} |S| + |U| + |U \times R| + |U \times H| \\ |H| + 3|U| + |U \times R \times H| \end{gathered}$$

in the worst-case.

V. Decompositions and Heuristics

In this section, two additional models are defined. First, Benders decomposition is used to further increase the scalability of the fog-UAV PLSCP without a loss in optimal cost. Second, a constraint-optimization model is defined to simply minimize the number of fog-UAVs and stations selected while satisfying the constraints.

A. Benders decomposition

Following the exact and relaxation variants of fog-UAV-MILP, we use Benders decomposition [28] to further increase the scalability of the fog-UAV PLSCP. Benders decomposition is most efficient when there is a discernible block structure of continuous variables.We use fog-UAV-MILP-R as a template, with relaxed variables and constraints outlined in IV-G. From the fog-UAV-MILP-R model, it is seen that 1) the optimal set of station variables x are dependent on fog-UAV variables y, and 2) the optimal set of dimensioning variables z are dependent on UAV-hotspot variables p. Hence, fog-UAV-MILP-R is reformulated as one problem with binary variables y and p, and two 'blocks' – one with variables y and x, and the other with variables p and z. That is,

$$(BD_0) \quad \text{Min} \quad \sum_{u \in U} y_u \cdot V_u + g_y(\mathbf{y}) + g_p(\mathbf{p}) \tag{33}$$

Subject to

$$\log(1-\alpha) \geq \sum_{u \in \mathcal{U}_h} p_u^h \log(\mathcal{B}_{u,h}) \tag{21}$$

$$\frac{1}{|\mathcal{H}_u|} \sum_{h \in \mathcal{H}_u} p_h^u \leq y_u; \quad u \in U \tag{29}$$

where

$$g_y(\mathbf{y}) = \text{Min} \quad \sum_{s \in S} x_s \cdot C_s \tag{34}$$

Subject to

$$x_{\xi(u)} = \tilde{y}_u; \quad u \in U \qquad (\varepsilon_u)$$

$$\sum_{s \in \mathcal{W}_u \setminus \xi(u)} x_s \geq (Y_u - 1) \cdot \tilde{y}_u; \quad u \in U \qquad (\varsigma_u)$$

and

$$g_p(\mathbf{p}) = \text{Min} \quad \sum_{u \in U} \sum_{r \in R} v_u^r \cdot z_u^r \tag{35}$$

$$\text{Subject to} \quad z_u^r \geq tT_h^u \Psi_{u,h}^r \tilde{p}_h^u, \qquad (\theta_{u,h}^r)$$

$$(r,u,h) \in R \times U \times H$$

for $\tilde{y}_u$ and $\tilde{p}_h^u$ fixed $\forall u \in U$, $\forall h \in H$.

1) Benders Subproblems: The Benders subproblems are defined by the duals of $g_y(\mathbf{y})$ and $g_p(\mathbf{p})$. First, recall that $\mathcal{W}_s$ is the set of viable fog-UAVs that can reach $s \in S$ (15). We define Ξ_s as the set of fog-UAVs that use s as a base station. That is,

$$\Xi_s = \{u \in U \mid \xi(u) = s\}, \quad s \in S. \tag{36}$$

Referring to the dual variables ε_u and ς_u corresponding to their appropriate constraints in $g_y(\mathbf{y})$ (34), the subproblem SP_y is defined as

$$(SP_y) \quad \text{Max} \quad \sum_{u \in U} \tilde{y} \cdot \varepsilon_u + \sum_{u \in U} (Y_u - 1)\tilde{y}_u \cdot \varsigma_u \tag{37}$$

$$\text{Subject to} \sum_{u \in \Xi_s} \varepsilon_u + \sum_{u \in \mathcal{W}_s \setminus \Xi_s} \varsigma_u \leq C_s \tag{38}$$

$$\varsigma_u \geq 0, \quad u \in U.$$

Similarly, referring to the dual variable $\theta_{u,h}^r$ corresponding to their appropriate constraints in $g_p(\mathbf{p})$ (35), the subproblem SP_p is defined as

$$(SP_p) \quad \text{Max} \quad \sum_{r \in R} \sum_{u \in U} \sum_{h \in H} tT_h^u \Psi_{u,h}^r \tilde{p}_h^u \cdot \theta_{u,h}^r \tag{39}$$

$$\text{Subject to} \quad \sum_{h \in H} \theta_{u,h}^r \leq v_u^r, \quad (r,u) \in R \times U \tag{40}$$

$$\theta_{u,h}^r \geq 0, \quad (r,u,h) \in R \times U \times H.$$

2) Master Problem: Based on the initial reformulation BD_0 (33) and the subproblems SP_y (37) and (39), the Benders Master Problem is defined as

$$(MP) \quad \text{Min} \quad \sum_{u \in U} V_u \cdot y_u + \mu + \nu \tag{41}$$

Subject to

$$\log(1-\alpha) \geq \sum_{u \in \mathcal{U}_h} p_u^h \log(\mathcal{B}_{u,h}) \tag{21}$$

$$\frac{1}{|\mathcal{H}_u|} \sum_{h \in \mathcal{H}_u} p_h^u \leq y_u; \quad u \in U \tag{29}$$

Optimality Cuts:

$$\sum_{u \in U} \tilde{y}_u \cdot (\varepsilon_u)^i + (Y_u - 1)\tilde{y}_u \cdot (\varsigma_u)^i \leq \mu \tag{42}$$

$$\sum_{r \in R} \sum_{u \in U} \sum_{h \in H} tT_h^u \Psi_{u,h}^r \tilde{p}_h^u \cdot (\theta_{u,h}^r)^i \leq \nu \tag{43}$$

$$\forall i = 1, 2, \ldots, I$$

Feasibility Cuts:

$$\sum_{u \in U} \tilde{y}_u \cdot (\varepsilon_u)^i + (Y_u - 1)\tilde{y}_u \cdot (\varsigma_u)^i \leq 0 \tag{44}$$

$$\sum_{r \in R} \sum_{u \in U} \sum_{h \in H} tT_h^u \Psi_{u,h}^r \tilde{p}_h^u \cdot (\theta_{u,h}^r)^j \leq 0 \tag{45}$$

$$\forall j = 1, 2, \ldots, J,$$

over I optimality cuts and J feasibility cuts. The master problem MP along with the subproblems SP_y and SP_p define the *fog-UAV-Benders* model.

3) Solving procedure: To solve fog-UAV-Benders, the master problem MP and subproblems SP_y and SP_p are solved in alternation. Each solution of MP defines an upper bound to the optimal solution. Each solution of the subproblems defines a lower bound to the optimal solution, and also generates appropriate optimality or feasibility cuts for MP. Fog-UAV-Benders terminates when the gap between the upper and lower bounds is sufficiently small [28].

B. Constraint optimization

A constraint optimization approach is defined to mimic a standard design and dimensioning approach. Known as *fog-UAV-CO*, the model is defined as

$$\text{Min} \quad C\sum_{s \in S} x_s + V \sum_{u \in U} y_u + \sum_{r \in R} v^r \Big(\sum_{u \in U} z_u^r \Big) \tag{46}$$

$$\text{Subject to } (21), (25), (28), (29), (30).$$

This model acts as a heuristic benchmark during the following simulations and analysis.

VI. Simulation & Analysis

In this section, we provide a detailed analysis of simulated results. Fog-UAV-MILP is an exact MILP from which all relaxations are derived. Therefore, it is used as the best-case solution cost benchmark and worst-case solution time benchmark.

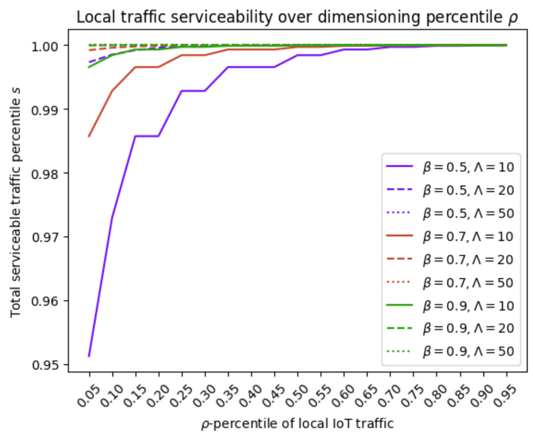


Figure 3: Serviceable local traffic within communication distance of a fog-UAV dimensioned for ρ-percentile traffic. Serviceability calculated over various stationary fog percentiles β and IoT request arrival rates Λ.

Section VI-A analyzes the traffic serviceability improvements achieved by the design & dimensioning of fog-UAVs. Section VI-B describes the simulation configurations and parameters. Section VI-C describes the computing environment and software on which the simulations were executed. Sections VI-D and VI-E analyze and compare the respective solution times and solution costs of the various models. Section VI-F examines the optimal design solutions for each map configuration. Finally, section VI-G summarizes the simulation results.

A. *Traffic Serviceability*

The improvements of fog-UAVs on IoT traffic serviceability can be observed over *local* IoT traffic and *global* IoT traffic.

1) Local IoT traffic: Consider a dense cluster of all IoT hotspots that are able to communicate, and hence be serviced, by a fog-UAV $u \in U$. That is, a cluster of hotspots within a δ_u-service radius. Let Λ_{δ_u} be the maximal arrival rate among all such IoT hotspot clusters. Let $T^{\beta}_{\mathcal{R}}$ be the β-percentile IoT traffic within an $\mathcal{R}$-radius cluster. The total percentile of serviceable traffic by u is s defined by

$$T^{\rho}_{\delta_u} = \mathcal{P}^{-1}(\rho;\ \Lambda_{\delta_u}) \tag{47}$$

$$s_u = \mathcal{P}(T^{\beta}_{\delta_u} + T^{\rho}_{\delta_u};\ \Lambda_{\delta_u}). \tag{48}$$

That is, the addition of a single fog-UAV dimensioned for ρ IoT traffic over a cluster with arrival Λ increases the traffic serviceability to its s-percentile.

Fig. 3 simulates (47) and (48) to show the total local traffic serviceability within the communication radius of a fog-UAV, dimensioned for ρ-percentile traffic, for various values of β and Λ. It can be seen that regardless of the β and Λ values graphed, a 95-percentile of local traffic can be serviced with as little as a 20-percentile dimensioning of fog-UAVs.

2) Global IoT traffic: High global IoT traffic is described by the number of distinct IoT clusters with IoT traffic that exceeds the β-percentile for any one resource $r \in R$.

Let τ_u be the ratio of the service radius over the total mobility radius of a fog-UAV u, considering additional stations and $\sim 80\%$ battery charge. That is,

$$N_u = (\kappa + \gamma) \cdot \eta_u \tag{49}$$

$$\tau_u = \frac{\delta_u}{N_u}, \tag{50}$$

where N_u represents the total mobility radius of u, and τ_u represents the fraction serviceable hotspots u can service at any one time.

For ϖ_α fog-UAVs within an N_u-radius area, the relative serviceability increase over the β-percentile traffic is

$$\chi_u(\alpha, \beta, \rho) = \frac{T^{\beta}_{N_u} + \varpi_{\alpha,\beta} \cdot T^{\rho}_{\delta_u}}{T^{\beta}_{N_u}}, \tag{51}$$

where $\varpi_{\alpha,\beta}$ is defined by

$$\varpi_{\alpha,\beta} = \left\lceil \frac{\log 1 - \alpha)}{\log(1-\beta)} \right\rceil, \tag{52}$$

which is a simplification of constraint (21).

Note that

$$T^{\beta}_{N_u} = \mathcal{P}^{-1}(\beta;\ \Lambda_{N_u}) \tag{53}$$

$$\begin{aligned} T^{\rho}_{\delta_u} &= \mathcal{P}^{-1}(\rho;\ \Lambda^{r}_{\delta_u}) \\ &= \mathcal{P}^{-1}(\rho;\ \tau_u \cdot \Lambda_{N_u}). \end{aligned} \tag{47}$$

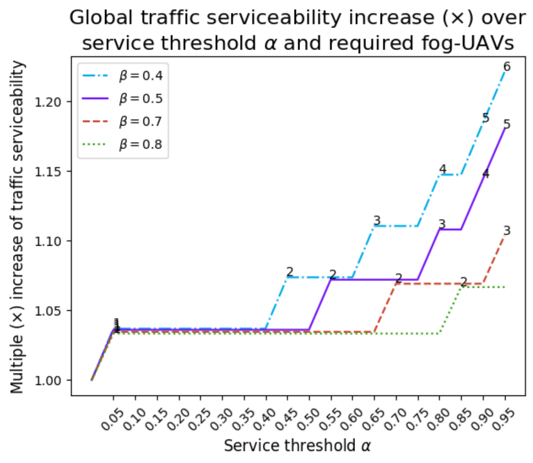
Hence, the added serviceability of the N_u-radius region is the ρ-percentile of a τ_u fraction of the region.

Equations (51) and (52) were simulated over increased values for α and a base traffic serviceability $\beta \in \{0.5, 0.7, 0.9\}$. Based on the local traffic serviceability results in Fig. 3, we let $\rho = 0.2$ to achieve a total 95-percentile local serviceability, based on the results in section VI-A1.

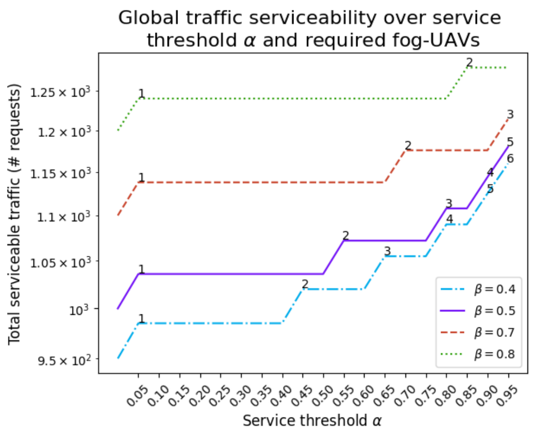
Fig. 4a shows the resulting simulation of the relative global traffic serviceability increase from the stationary fog infrastructure serviceability of β. It can be seen that a lower α with a lower β can achieve a similar relative serviceability improvements gain as a higher α with a higher β. For example, the (α, β) pair $(0.45, 0.4)$ and $(0.95, 0.9)$ both achieve a relative traffic increase of $\sim 1.07\times$.

Note, that α determines how many fog-UAVs are required for a high percentage of network serviceability. By Table IV, the relative increase in serviceability per fog-UAV is marginal for lower β. Indeed, a lower β means each fog-UAV with the same resource dimensioning will have a higher impact on the network serviceability than with higher β; however, the low variance of the Poisson distribution contributes to the small relative difference between the serviceability increases for various serviceability percentiles β.

Given that the relative serviceability increases are similar, the larger contributor to design & dimensioning cost becomes the number of required required fog-UAVs. Fig. 4b shows the absolute serviceability increase for differing β and number of fog-UAVs. For example, a fog-infrastructure with base service percentile of $\beta = 0.4$ requires 5 fog-UAVs to achieve the same serviceability as an infrastructure with base service percentile $\beta = 0.7$ and one fog-UAV. A fog infrastructure with $\beta = 0.4, 0.5, 0.7$ achieve similar total serviceability with



(a) Multiple $(\times)$ increase



(b) Absolute increase

Figure 4: Traffic serviceability increase by fog-UAVs over the β-percentile support of a stationary fog infrastructure, with the required number of fog-UAVs over the data points.

Table IV: Relative serviceability increase $(\times)$ per additional fog-UAV in Fig. 4a

β-**percentile serviceability**	0.4	0.5	0.7	0.9
Serviceability increase $(\times)$ **per fog-UAV**	1.0368	1.36	1.0346	1.0323

6, 5 and 3 fog-UAVs respectively. Hence, there is a trade-off between the design & dimensioning of the stationary fog infrastructure, and the mobile fog-UAV infrastructure. The optimal balance of serviceability between the stationary and mobile fog infrastructures will be studied in future work.

B. Simulation Configurations

The size and hotspot density of the two maps used for simulation are meant to loosely approximate the size and density of Montreal and Toronto, two large and densely populated cities. By Statistics Canada, the populations of Montreal and Toronto were ~$1,794,000$ and ~$2,857,000$ people respectively in 2021 with 95% confidence, over areas of ~365 km^2 and ~631 km^2 respectively [29]. As each hotspot represents ~$10,000$ people,

Class	Travel Distance	Radius	Resource Capacity
Class 0:	3 km	0.5 km	(2.5 GHz, 4 GB, 8 GB)
Class 1:	8 km	1 km	(5 GHz, 8 GB, 16 GB)
Class 2:	12 km	2 km	(7.5 GHz, 16 GB, 64 GB)

Table V: Fog-UAV configurations for simulation.

two maps were created with 150 and 250 hotspots respectively. Furthermore, historic analysis of international cities show that most cities are approximately circular or elliptical, with a higher population density at their center [30]. Hence, each map was created by uniformly placing hotspots over a 25×25 unit region with polar coordinates within the ellipse

$$(0.85x)^2 + (1.1y)^2 = 12.5^2.$$

Such an ellipse adds variety in the hotspot density along the city's length and width, and has an area of 525 km^2. Figure 5 depicts the two hotspot maps of different population densities used for simulation. The population densities of these two maps are ~2850 km^2 and ~4750 km^2.

Each hotspot $h \in H$ with location $(h_{\mathrm{x}}, h_{\mathrm{y}})$ has an hourly request arrival rate of $h = 100$, which yields high local IoT serviceability for a low ρ-dimensioning (Fig. VI-A1). Each hotspot also corresponds to a station candidate $s \in S$ with same location $(h_{\mathrm{x}}, h_{\mathrm{y}})$ and uniform random station cost $C_s \in [500, 1000]$. Each station is the base station to 3 *classes* of fog-UAV, each with a set *travel distance* η_u and *radius* δ_u in kilometres (km), and a *resource capacity* $(Z_u^r)_{r=1}^3$ triplet for resources $r \in R$. This was done to mimic the variety and power of different UAVs on the market today [31]. The configuration details of the UAV classes are available in Table V. The travel distance and radius of each fog-UAV class is represented over the maps in Fig. 5. The mobility fractions (κ, γ) for hotspot and station reachability were set to $(0.4, 0.8)$ to ensure fog-UAVs could move between stations and outreach to hotspots without worrying about battery failures. Fog-UAV CAPEX costs were set to $V_u \in \{\$250, \$500, \$750\}$ depending on class, and per unit resource costs were set randomly to $v_u^r \in [\$50, \$100]$.

Each iteration i of the simulation casts a service area covering $2i$ km^2. Let H_i be the set of hotspots serviced in iteration i. The total demand (i.e., number of requests) over simulation i is

$$d_i = \lambda(1-\beta)\cdot|H_i| \tag{54}$$

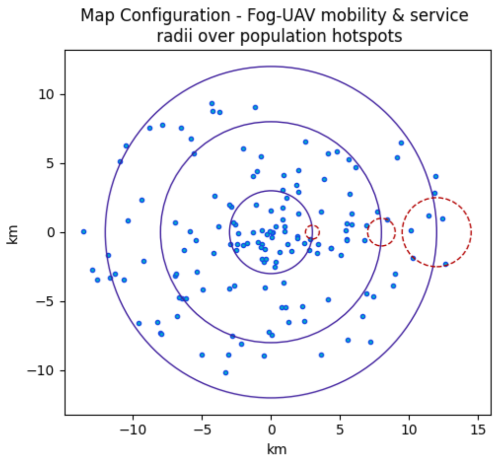
$$D_i \sim Poisson(d_i). \tag{55}$$

The mean demand d_i becomes the x-axis of the solution graphs in Fig. 6 and 7.

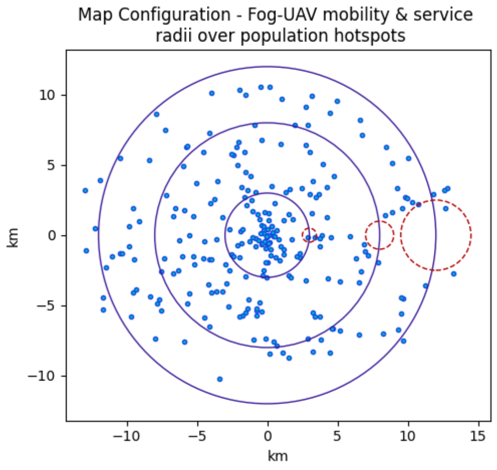
The simulations execute each of the four defined models: fog-UAV-MILP, fog-UAV-MILP-R, fog-UAV-B, and fog-UAV-CO. Each simulation iteration $i \in \mathbb{N}^{\geq 1}$ is executed over an increasing subset of hotspots, candidate stations and candidate fog-UAVs. Each iteration is executed 3 times, and the average is used for analysis purposes.

Based on analysis of traffic serviceability in VI-A, simulations were executed for parameter values (β, α) in

(a) Legend


Map Configuration - Fog-UAV mobility & service radii over population hotspots
km
km

(b) Density ~2850 people/km^2


Map Configuration - Fog-UAV mobility & service radii over population hotspots
km
km

(c) Density ~4750 people/km^2

Figure 5: Map configurations used for simulations, representing the hotspot placements and fog-UAV mobility & service radii defined in VI-B.

$\{(0.7, 0.95), (0.8, 0.5)\}$ to achieve a high percentage of total serviceability, and for each simulation map of densities ~2850 km^2 and ~4750 km^2 in Fig. 5. That is, two parameter pairs and two simulation maps of different population densities yielded four sets of simulations and sets of results.

C. Environmental Configurations

All simulations were executed with IBM CPLEX optimization software in C++. Simulations were executed on a machine with 24GB of RAM and an Intel i5-7500 CPU @ 3.40 GHz.

D. Solution Time

Solution times in seconds of each model are shown over increasing demand in Fig. 6 for parameters $\beta = 0.7,\ \alpha = 0.95$, and in Fig. 7 for parameters $\beta = 0.8,\ \alpha = 0.5$. From the graphs, it can be seen that Fog-UAV-MILP executes significantly slower than the other models, whereas Fog-UAV-CO, the heuristic, executes quicker as demand grows. Both Fog-UAV-MILP-R and Fog-UAV-Benders remain scalable over increasing demand, with Fog-UAV-Benders outperforming Fog-UAV-MILP-R on average by 11-52%. Though this decrease in execution time varies significantly, Fig. 6 and 7 show this decrease is consistent over time.

Table VI: Analysis of execution time increases between various model pairs, for varying α, β and map density.

(a) $\beta = 0.7,\ \alpha = 0.95$, density ~2850 people/km^2

Base Model	Improved Model	Improvement Mean	Std. Dev.
Fog-UAV-MILP	Fog-UAV-MILP-R	5.233×	3.154×
Fog-UAV-MILP	Fog-UAV-Benders	6.703×	4.448×
Fog-UAV-MILP-R	Fog-UAV-Benders	1.238×	0.118×
Fog-UAV-Benders	Fog-UAV-CO	5.793×	135.99×

(b) $\beta = 0.7,\ \alpha = 0.95$, density ~4750 people/km^2

Fog-UAV-MILP	Fog-UAV-MILP-R	13.9485×	0.7996×
Fog-UAV-MILP	Fog-UAV-Benders	17.025×	6.868×
Fog-UAV-MILP-R	Fog-UAV-Benders	1.5221×	0.347×
Fog-UAV-Benders	Fog-UAV-CO	0.291×	0.0344×

(c) $\beta = 0.8,\ \alpha = 0.5$, density ~2850 people/km^2

Fog-UAV-MILP	Fog-UAV-MILP-R	2.51×	1.16×
Fog-UAV-MILP	Fog-UAV-Benders	3.03×	0.693×
Fog-UAV-MILP-R	Fog-UAV-Benders	1.21×	0.34×
Fog-UAV-Benders	Fog-UAV-CO	7.425×	68.54×

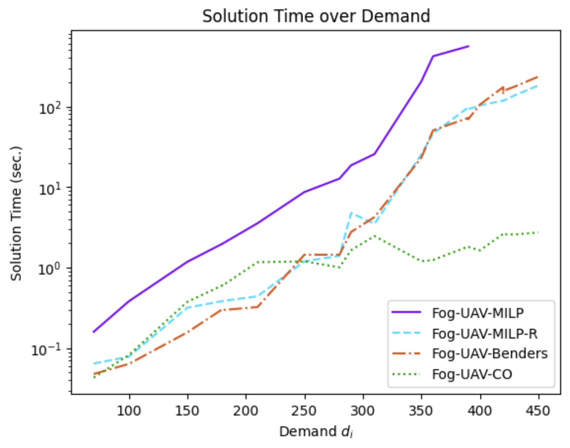
(d) $\beta = 0.8,\ \alpha = 0.5$, density ~4750 people/km^2

Fog-UAV-MILP	Fog-UAV-MILP-R	7.52×	0.758×
Fog-UAV-MILP	Fog-UAV-Benders	7.982×	1.96×
Fog-UAV-MILP-R	Fog-UAV-Benders	1.11×	0.035×
Fog-UAV-Benders	Fog-UAV-CO	1.08×	0.1935×

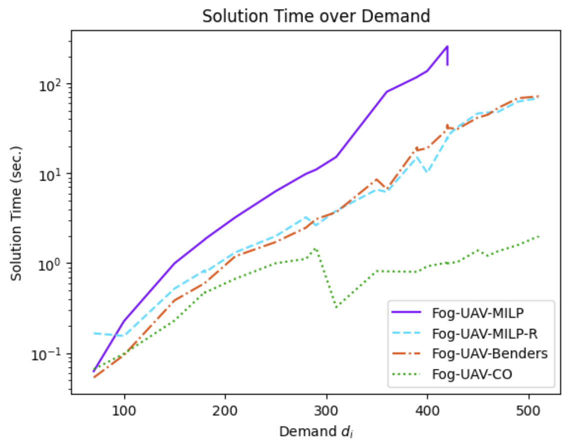
A numerical analysis of the improvements in execution time between various models is outlined in Table VI for each of the four parameter-map pairings. Indeed, Fog-UAV-Benders executes on average as low as 3.03× faster than Fog-UAV-MILP for $\beta = 0.8,\ \alpha = 0.5$ and density ~2850 km^2, and as much as 17.025× faster for $\beta = 0.7,\ \alpha = 0.95$ and density ~4750 km^2. The solution time improvement of Fog-UAV-Benders over Fog-UAV-MILP increases over higher density maps, given the same parameter pair (β, α). Conversely, though Fog-UAV-CO executed more quickly on average than Fog-UAV-Benders, the improvement in solution time decreased as map density increases.

E. Solution Cost

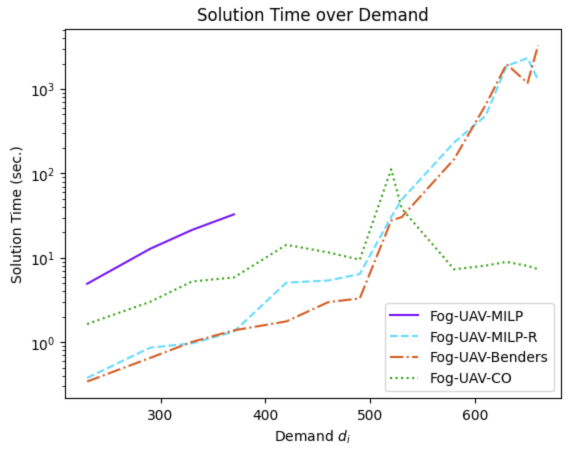
A numerical analysis of the increase in solution cost between various models is outlined in Table VII for each of the four parameter-map pairings. Over all simulation iterations, the cost of Fog-UAV-MILP, Fog-UAV-MILP-R and Fog-UAV-Benders are identical. Since Fog-UAV-MILP executes signif-



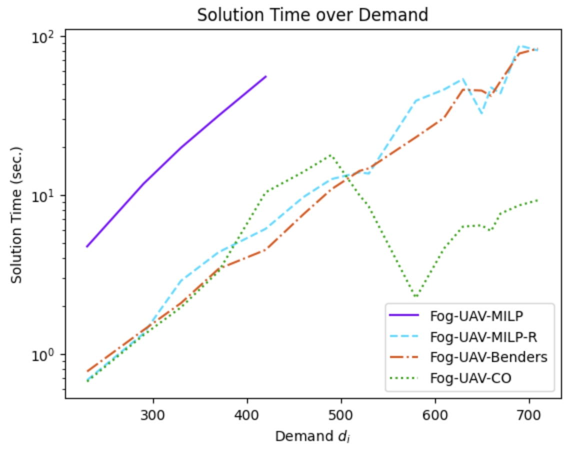
(a) $\beta = 0.7,\ \alpha = 0.95$, density ~2850 people/km^2



(a) $\beta = 0.8$, $\alpha = 0.5$, density ~2850 people/km^2



(b) $\beta = 0.7,\ \alpha = 0.95$, density ~4750 people/km^2



(b) $\beta = 0.8$, $\alpha = 0.5$, density ~4750 people/km^2

Figure 6: Solution time of each model for varying map density; $\beta = 0.7,\ \alpha = 0.95$

Figure 7: Solution time of each model for varying map density; $\beta = 0.7$, $\alpha = 0.5$

icantly slower than both Fog-UAV-MILP-R and Fog-UAV-Benders with identical cost, it is always preferable to use one of the two relaxed models.

On average over the four parameter-map pairings, Fog-MILP-CO has a solution cost between 8.17% and 13.65% worse than the exact solution, which may be viewed as a significant increase.

F. Design Solution

The optimal design solutions over a map with population density ~4750km^2 is shown in Fig. 8. Based on the service constraint (21), the (β, α) pair $(0.7, 0.95)$ is expected to require more fog-UAVs per hotspot than the pair $(0.8, 0.5)$. Indeed, the optimal solution of pair $(0.7, 0.95)$ in Fig. 8b requires more fog-UAVs and charging stations than the optimal solution of pair $(0.8, 0.5)$ in Fig. 8c. Interestingly, the optimal solution is composed primarily of lower cost fog-UAVs with lower mobility and service radius. As expected, charging stations are clustered towards the densest portion of the map. This implies there is a reliance on the mobility of fog-UAVs to service the demand outside the city center. This may also suggest that the majority of traffic spikes are near the city centers.

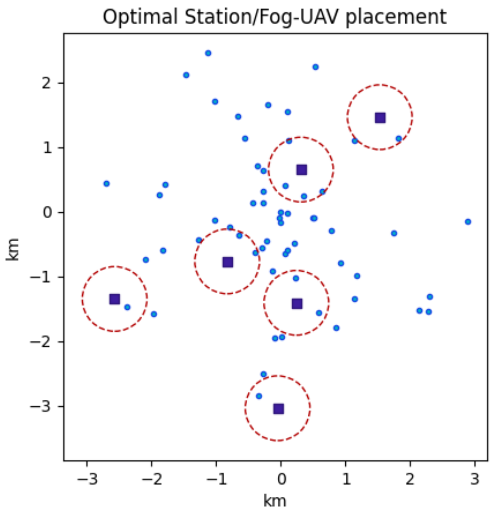
G. Summary of Results

The continuous relaxation of certain binary variables in fog-UAV-MILP-R and the precomputation of viability constraints were chosen to guarantee an equivalent cost as fog-UAV-MILP while decreasing the model complexity. As seen in Table VII, this hypothesis holds as fog-UAV-MILP-R yields an equivalent solution cost. By using Benders decomposition, we decreased the complexity and execution time of fog-UAV-Benders by ~11-52%, while maintaining the optimal cost of fog-UAV-MILP. Therefore, it is reasonable to use fog-UAV-Benders for a large scale fog-UAV PLSCP.

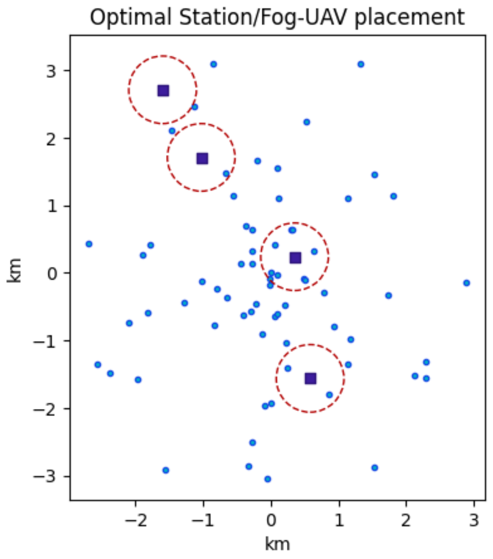
VII. Conclusion & Future Work

In this paper, we designed & dimensioned a fog-UAV PLSCP to supplement a fixed fog infrastructure and support overloaded hotspots. The design & dimension of a fog-UAV set covering is optimized to have 1) flexible UAV deployment, 2) additional fog-UAVs per station for backup in case of a

(a) Legend



(b) $\beta = 0.7$, $\alpha = 0.95$ over 30 km^2 area



(c) $\beta = 0.8$, $\alpha = 0.5$ over 36 km^2 area

Figure 8: Optimal station and fog-UAV placements for map density of ~4750 people/km^2 over varying α and β and coverage area. Fog-UAV service radius is depicted over its associated base station.

Table VII: Analysis of relative cost differences between the exact solution and others, for varying α, β and map density.

(a) $\beta = 0.7$, $\alpha = 0.95$, density ~2850 people/km^2

Exact Model	**Heuristic Model**	**Increase Mean**	**Increase Std. Dev.**
Fog-UAV-MILP	Fog-UAV-MILP-R	0.0%	0.0%
Fog-UAV-MILP-R	Fog-UAV-B	0.0%	0.0%
Fog-UAV-MILP-B	Fog-MILP-CO	10.06%	0.858%

(b) $\beta = 0.7$, $\alpha = 0.95$, density ~4750 people/km^2

Fog-UAV-MILP	Fog-UAV-MILP-R	0.0%	0.0%
Fog-UAV-MILP-R	Fog-UAV-B	0.0%	0.0%
Fog-UAV-MILP-B	Fog-MILP-CO	6.58%	0.0476%

(c) $\beta = 0.8$, $\alpha = 0.5$, density ~2850 people/km^2

Fog-UAV-MILP	Fog-UAV-MILP-R	0.0%	0.0%
Fog-UAV-MILP-R	Fog-UAV-B	0.0%	0.0%
Fog-UAV-MILP-B	Fog-MILP-CO	9.24%	0.18%

(d) $\beta = 0.8$, $\alpha = 0.5$, density ~4750 people/km^2

Fog-UAV-MILP	Fog-UAV-MILP-R	0.0%	0.0%
Fog-UAV-MILP-R	Fog-UAV-B	0.0%	0.0%
Fog-UAV-MILP-B	Fog-MILP-CO	8.17%	0.24%

depleted battery, 3) untethered mobility of fog-UAVs, and 4) minimal latency of UAV deployment.

The MILP models were designed with chance constraints to derive reasonable fog-UAV service in extreme cases of severely high traffic. Constraint precomputation and variable relaxations were used to reduce the complexity of the MILP model. Benders decomposition was used to further reduce the complexity of the relaxed MILP model. In-depth analysis of simulated execution time and cost show that the fog-UAV-Benders model is significantly faster than the MILP variants, while maintaining the optimal design & dimensioning cost.

This paper makes simplifications and assumptions for the sake of a concise model that can be expanded in future work. Firstly, our definition of 'high IoT traffic' is limited to the number of requests from each hotspots. Future models will consider high traffic both in the number of requests, and in the resource requirements of the requests.

Our model ensures that each fog-UAV has multiple reachable charging stations. We assume that all deployed fog-UAVs are at least 80% charged, and each station can store all reachable fog-UAVs. We will use a stochastic programming models [32] to find optimal fog-UAV partial charging and re-distribution policies among charging stations. Larger scale discrete event simulations [33] will be used to evaluate the real-time performance of resulting solutions.

Our model assumes a one-hop unhindered communication between fog-UAVs and hotspots within a fog-UAV's service radius. That is, forms of communication interference such as buildings, natural terrain or other communication networks are not considered [34]. In addition, communication strength for varying fog-UAV hovering heights are not considered. In future work, we will seek to construct a Proof-of-Concept fog-UAV model to explore the effects of terrain and distance on the strength and consistency of fog-UAV communication with IoT.

Based on simulations over the IoT traffic serviceability increase with fog-UAVs given a base stationary fog infrastructure, it can be seen that there is a significant trade-off between the size of the stationary and mobile fog infrastructures to achieve an equivalent traffic serviceability. Hence, once a better understanding of IoT traffic and UAV communication interference is obtained, we will design an end-to-end mobile of fog design & dimensioning which include both stationary and mobile fog elements.

Furthermore, we will construct a fog infrastructure that is composed of fixed fog nodes [7], on-demand fog nodes [8], and mobile fog-UAVs. Using this fog infrastructure over a city network of hotspots, we will simulate the activation and support of fog nodes to varying IoT requests over time, and factoring both flight time, discharge time [35] and charging time [27]. In doing so, we can quantify the significance of the proposed fog infrastructure, identify outstanding bottlenecks in the system, and move closer towards a reliable real-time fog infrastructure.

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [PGSD3 - 558695 - 2021].

References

[1] S. K. Lee, M. Bae, and H. Kim, "Future of iot networks: A survey," *Applied Sciences*, vol. 7, no. 10, p. 1072, 2017.

[2] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn, *et al.*, "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures," *Journal of Cloud Computing*, vol. 7, no. 1, p. 16, 2018.

[3] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149–160, 2019.

[4] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.

[5] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management and evaluation of fog computing systems: A survey," *IEEE Internet of Things Journal*, 2020.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[7] I. Martinez, A. Jarray, and A. S. Hafid, "Scalable design and dimensioning of fog-computing infrastructure to support latency sensitive iot applications," *IEEE Internet of Things Journal*, 2020.

[8] I. Martinez, A. S. Hafid, and M. Gendreau, "Robust and fault-tolerant fog design & dimensioning for reliable operation," *IEEE Internet of Things Journal*, 2022.

[9] A. J. Jara, D. Genoud, and Y. Bocchi, "Big data in smart cities: from poisson to human dynamics," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pp. 785–790, IEEE, 2014.

[10] C. ReVelle and K. Hogan, "A reliability-constrained siting model with local estimates of busy fractions," *Environment and Planning B: Planning and Design*, vol. 15, no. 2, pp. 143–152, 1988.

[11] C. Revelle and K. Hogan, "The maximum reliability location problem and α-reliable p-center problem: derivatives of the probabilistic location set covering problem," *Annals of Operations Research*, vol. 18, no. 1, pp. 155–173, 1989.

[12] A. Fakhri, M. Ghatee, A. Fragkogios, and G. K. Saharidis, "Benders decomposition with integer subproblem," *Expert Systems with Applications*, vol. 89, pp. 20–30, 2017.

[13] S. Chapman and J. White, "Probabilistic formulations of emergency service facilities location problems," in *ORSA/TIMS Conference, San Juan, Puerto Rico*, 1974.

[14] P. Beraldi and A. Ruszczyński, "The probabilistic set-covering problem," *Operations Research*, vol. 50, no. 6, pp. 956–967, 2002.

[15] C. ReVelle, C. Toregas, and L. Falkson, "Applications of the location set-covering problem," *Geographical analysis*, vol. 8, no. 1, pp. 65–76, 1976.

[16] H. K. Rajagopalan, C. Saydam, and J. Xiao, "A multiperiod set covering location model for dynamic redeployment of ambulances," *Computers & Operations Research*, vol. 35, no. 3, pp. 814–826, 2008.

[17] A. Shariat-Mohaymany, M. Babaei, S. Moadi, and S. M. Amiripour, "Linear upper-bound unavailability set covering models for locating ambulances: Application to tehran rural roads," *European Journal of Operational Research*, vol. 221, no. 1, pp. 263–272, 2012.

[18] N. Madan, A. W. Malik, A. U. Rahman, and S. D. Ravana, "On-demand resource provisioning for vehicular networks using flying fog," *Vehicular Communications*, p. 100252, 2020.

[19] K. G. Panda, S. Das, D. Sen, and W. Arif, "Design and deployment of uav-aided post-disaster emergency network," *IEEE Access*, vol. 7, pp. 102985–102999, 2019.

[20] F. Zhou, Y. Wu, H. Sun, and Z. Chu, "Uav-enabled mobile edge computing: Offloading optimization and trajectory design," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[21] Q. Wu, L. Liu, and R. Zhang, "Fundamental trade-offs in communication and trajectory design for uav-enabled wireless network," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 36–44, 2019.

[22] L. Chiaraviglio, L. Amorosi, N. Blefari-Melazzi, P. Dell'Olmo, C. Natalino, and P. Monti, "Optimal design of 5g networks in rural zones with uavs, optical rings, solar panels and batteries," in *2018 20th international conference on transparent optical networks (icton)*, pp. 1–4, IEEE, 2018.

[23] Y. Park, P. Nielsen, and I. Moon, "Unmanned aerial vehicle set covering problem considering fixed-radius coverage constraint," *Computers & Operations Research*, vol. 119, p. 104936, 2020.

[24] R. Mao, B. Du, D. Sun, and N. Kong, "Optimizing a uav-based emergency medical service network for trauma injury patients," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 721–726, IEEE, 2019.

[25] K. Tutschku and P. Tran-Gia, "Spatial traffic estimation and characterization for mobile communication network design," *IEEE Journal on selected areas in communications*, vol. 16, no. 5, pp. 804–811, 1998.

[26] "Statistics Canada. 2023. (table). census profile. 2021 census of population. Statistics Canada catalogue no. 98-316-x2021001. ottawa. released march 29, 2023.." https://www12.statcan.gc.ca/census-recensement/2021/dp-pd/prof/index.cfm? Accessed: 2023-05-17.

[27] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, "The electric vehicle routing problem with nonlinear charging function," *Transportation Research Part B: Methodological*, vol. 103, pp. 87–110, 2017.

[28] Z. C. Taskın, "Benders decomposition," *Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Malden (MA)*, 2010.

[29] Statistics Canada, "Census of Population - 2021." https://www12.statcan.gc.ca/census-recensement/index-eng.cfm, 2022. [Online; accessed 9-April-2024].

[30] M. Batty and P. Longley, "The shape of cities: geometry, morphology, complexity and form," *Fractal Cities: A Geometry of Form and Function*, pp. 7–57, 1994.

[31] U. Khan, "How much is a drone? cost of different types of drones." https://www.linkedin.com/pulse/how-much-drone-cost-different-types-drones-usman-khan-5ovff/, 2024. [Online; accessed 9-April-2024].

[32] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[33] J. Misra, "Distributed discrete-event simulation," *ACM Computing Surveys (CSUR)*, vol. 18, no. 1, pp. 39–65, 1986.

[34] W. Mei, Q. Wu, and R. Zhang, "Cellular-connected uav: Uplink association, power control and interference coordination," *IEEE Transactions on wireless communications*, vol. 18, no. 11, pp. 5380–5393, 2019.

[35] C. M. Shepherd, "Design of primary and secondary cells: Ii. an equation describing battery discharge," *Journal of the Electrochemical Society*, vol. 112, no. 7, p. 657, 1965.

Chapter 6

A Blockchain-Based Audit Mechanism for Trust and Integrity in IoT-Fog Environments

Abstract The full realization of smart city technology is dependent on the secure and honest collaboration between IoT applications and edge-computing. In particular, resource constrained IoT devices may rely on fog-computing to alleviate the computing load of IoT tasks. Mutual authentication is needed between IoT and fog to preserve IoT data security, and monetization of fog services is needed to promote the fog service ecosystem. However, there is no guarantee that fog nodes will always respond to IoT requests correctly, either intentionally or accidentally. In the public decentralized IoT-fog environment, it is crucial to enforce integrity among fog nodes. In this paper, we propose a blockchain-based system that 1) streamlines the mutual authentication service monetization between IoT and fog, 2) verifies the integrity of fog nodes via service audits, and 3) discourages malicious activity and promotes honesty among fog nodes through incentives and penalties.

Previous chapters enable the implementation of a scalable, fault-tolerant fog infrastructure in a dynamic IoT-fog environment. This chapter defines a blockchain-based system to enhance the security and integrity between IoT and fog, and to facilitate fog service monetization. This chapter has been submitted to IEEE Transactions on Industrial Informatics [42].

A Blockchain-Based Audit Mechanism for Trust and Integrity in IoT-Fog Environments

Ismael Martinez, Abdelhakim Senhaji Hafid, Michel Gendreau

***Abstract*—The full realization of smart city technology is dependent on the secure and honest collaboration between IoT applications and edge-computing. In particular, resource constrained IoT devices may rely on fog-computing to alleviate the computing load of IoT tasks. Mutual authentication is needed between IoT and fog to preserve IoT data security, and monetization of fog services is needed to promote the fog service ecosystem. However, there is no guarantee that fog nodes will always respond to IoT requests correctly, either intentionally or accidentally. In the public decentralized IoT-fog environment, it is crucial to enforce integrity among fog nodes. In this paper, we propose a blockchain-based system that 1) streamlines the mutual authentication service monetization between IoT and fog, 2) verifies the integrity of fog nodes via service audits, and 3) discourages malicious activity and promotes honesty among fog nodes through incentives and penalties.**

***Index terms*—** Internet of Things, fog computing, blockchain, service auditing, mutual authentication, smart contracts

I. Introduction

The *Internet of Things* (IoT) is an ever growing paradigm of sensors and computing devices inter-connected through the internet. IoT has emerged in both public and private sectors with the main objective of facilitating our lives [1]. A wide scale network of collaborative IoT applications is the first step towards the implementation of smart cities [2].

Many IoT devices and applications rely on external computation and storage due to limited internal resources. Though Cloud data-centers are heavily equipped to support any number of IoT requests, network congestion near distant Cloud data-centers may result in high response latency to IoT devices [3]. This high latency can be an inhibiting factor for certain real-time IoT applications in health care [4], autonomous vehicles [5], and multimedia [6].

Fog-computing is a computational extension of Cloud services to the edge of the network. The fog layer is composed of geographically distributed 'micro data-centers', or nodes, that are positioned to support IoT with minimal latency [7]. Indeed, fog, alongside IoT and Cloud, are integral in creating an energy-efficient network computing architecture for smart cities [8], [9], [10].

I. Martinez is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ismael.martinez@umontreal.ca).

A. S. Hafid is with the Department of Computer Science and Operations Research, University of Montreal, Quebec, Canada H3C 3J7 (e-mail: ahafid@iro.umontreal.ca).

M. Gendreau is with the Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Quebec, H3C 3A7, Canada (email:michel.gendreau@polymtl.ca)

Current research in fog-computing focuses on the effective design of fog infrastructures [10], and resource off-loading policies from IoT to fog nodes [11]. However, such research does not consider the mutual needs of IoT and fog. IoT devices require real-time, secure and correct service from an authenticated server. Fog nodes require payment and advertisement for services from authenticated sources.

Furthermore, current work assumes that fog nodes always behave with *integrity*. That is, IoT devices are meant to blindly trust fog nodes even though it is possible that a fog node returns a faulty response, either intentionally or accidentally [12]. Indeed, the IoT-fog environment is **trustless** and currently lacks accountability for fog nodes to behave correctly. If IoT devices rely on fog nodes for computational processing, it is critical that we ensure active fog nodes are processing correctly, and eject malicious fog nodes from the IoT-fog environment.

In addition, IoT networks can be easy to tamper with and compromise without proper security measures. Blockchain technologies have been studied as a possible solution to provide security, privacy and access control to IoT due to the decentralization, immutability and high transparency of blockchain [13]. Hence, blockchain can be used to provide a secure line of communication between IoT and fog via mutual authentication [14]. Furthermore, blockchain can streamline the payment process from IoT for fog computing services, and enforce integrity among the fog nodes.

Based on observations of IoT-fog requirements, and limitations of current work, there exists a need for a single streamlined system in a trustless IoT-fog environment that 1) mutually authenticates IoT and fog prior to service, 2) facilitates service payment from IoT to fog, 3) verifies and holds malicious fog nodes accountable, and 4) benefits honesty and discourages malicious activity among fog nodes.

Inspired by current data auditing techniques [15], we propose a service auditing process for fog-computing to enforce computational integrity. To the best of our knowledge, this is the first attempt to enforce the service integrity of fog via a service auditing scheme. We also integrate current mutual authentication [14], [16] and fog monetization schemes [17], [18] into a single blockchain application, and leverage blockchain-enabled fog nodes to decrease latency [19]. That is, we propose the *Fog Identity & Service Integrity Enforcement* (FISIE) system that streamlines IoT-fog authentication, service, monetization, and integrity auditing through a single smart contract. The FISIE smart contract described in this paper is a Proof-of-Concept based on Ethereum [20]. However, any other smart-contract capable blockchain platform would be compatible with this system.

Our contributions are as follows:

- We review and summarize current literature related to payment, service and mutual authentication.
- We propose a general architecture of heterogeneous IoT and blockchain-enabled fog that is compatible with any smart contract-enabled blockchain.
- We define a smart contract-based system for mutual authentication, monetization, and service auditing.
- We describe a penality system to enforce service integrity.
- We discuss the security of the system, and analyze various auditing scheduling policies for optimal system integrity.

The remainder of this paper is organized as follows. Section II reviews the current contributions in related fields to inspire our solution. Section III provides an overview of the different components of the FISIE system. Section IV provides background knowledge and configurations specifics of blockchain, cryptography, and the IoT-fog physical layers. Section V initializes the smart contract. Sections VI, VII and VIII respectively define the identity management, payment management and integrity verification functions of the smart contract. Section IX describes how the smart contract functions provide penalties and incentives for fog integrity. Section X discusses the FISIE system security, and analyses the affects of different sampling policies on long-term fog integrity. Finally, section XI summarizes future work and concludes the paper.

II. Related Work

We are interested in providing security and integrity to the IoT-fog environment without significantly increasing communication latency. Our reviewed literature focuses on the state-of-the-art in a) IoT-fog security, b) data auditing of fog, c) blockchain-based monetization, and d) blockchain-fog integration.

A. IoT-fog security

Two of the key elements in providing security to any system is the inclusion of authorization & authentication [13]. In particular for the IoT-fog environment, we review implementations of access control for IoT data, and mutual authentication between IoT and fog.

1) Authorization: An *access control* policy defines which entities have the authority to access the data of which devices. Access control policies may list individual valid entities, or list attributes that entities must have to gain access [21]. A micro server such as fog has sufficient storage and computing resources to define and validate its own access control policy. However, IoT devices have minimal resources, and may not be able to store its own list of valid entities. In this case, the IoT access control policies are stored in a separate trusted server with sufficient resources.

Algarni et al. [22] propose a blockchain-based access control scheme for IoT. This scheme takes advantage of the transparency and security of blockchain to house all IoT access control policies. Since the blockchain itself cannot be hosted on the IoT devices, the fog layer can be used to host the blockchain and decrease communication latency between the blockchain and IoT.

2) Authentication: The authorization process often works in tandem with an *authentication* mechanism to prove the identity of a communicating entity [21]. The authentication process is crucial in protecting IoT and fog from security risks such as man-in-the-middle attacks and replay attacks [23]. Secure authentication in IoT, fog and Cloud are often based in standard encryption schemes such as RSA or elliptic curve cryptography (ECC), though ECC is known to be more secure than RSA for equivalent key sizes [24]. In the IoT-fog environment, we are interested in *mutual authentication*, wherein an IoT device and a fog node authenticate each other prior to communicating and data sharing [25].

Singh and Chaurasiya [25] propose a lightweight mutual authentication scheme with a centralized Cloud data-center as a trusted third-party. Based on ECC, the Cloud data-center sets all relevant cryptographic parameters, while IoT devices and fog nodes store only their own public keys. That is, private keys are stored on Cloud instead of the IoT/fog devices. Though this scheme is lightweight, storing minimal data on IoT devices, it requires absolute trust in the Cloud data-center. In addition, requiring communication with the Cloud increases communication latency for IoT.

Instead, we consider the use of a decentralized blockchain for the authentication process. Current contributions [14], [16], [26] use a smart contract-based scheme to register or remove identification information from IoT devices and fog nodes. Once registered, the information is stored and queried from a trusted off-chain table. However, these schemes rely on an additional centralized registration authority to generate and store keys for IoT devices and fog nodes [14], [16]. Giving a centralized authority this level of control over the system's private keys is a potential security risk. Instead, we propose to limit the use of any off-chain resources, and keep all private keys on their respective devices.

Patwary et al. [26] propose a blockchain-based mutual authentication scheme that uses the physical fog location data as part of the authentication process. Though the use of centralized resources are limited, this scheme only works with stationary IoT devices and fog nodes since authentication relies on a static location validation. Instead, we seek to implement a generalized authentication scheme that allows for device mobility without compromising IoT-fog security.

B. Data auditing of Fog

Several contributions propose a similar data auditing scheme to verify the data replica cache of edge servers [12], [15], [27]. A vendor who has previously cached its own data to edge servers may request the hash of the data replica from edge servers. The vendor compares the hash with its own data hash to verify the data integrity of an edge server.

Zikratov et al. [28] propose a data auditing scheme based on a private blockchain. Data is distributed to clients and is also stored on the blockchain. Periodically, the client data is downloaded and verified with the blockchain data by a third party auditor.

Tian et al. [29] address the problem of data auditing in a public IoT-fog environment. They propose to tag IoT data which is sent to a fog node which places its own tag, and then sends it to the Cloud. A third party auditor can then verify the integrity of the fog nodes via a zero-knowledge proof of integrity.

In both cases [28], [29], absolute cooperation is needed from fog nodes to honestly share or allow access to its server data. This level of trust cannot be guaranteed in a trustless system.

C. Blockchain-based Monetization

Service payments from IoT to fog have been previously considered by means of blockchain smart contracts [17], [18]. Debe et al. [17] consider a monetization smart contract in which IoT devices deposit Ether, which are then used to pay for fog services. Huang et al. [18] use a smart contract to hold a collateral deposit from IoT until the IoT device directly pays the fog node. If payment is not processed in a timely manner, the collateral is given to the fog node.

The system by Huang et al. [18] uses a commitment-based sampling approach in which the IoT devices samples a portion of the result from the fog node, to decide whether to pay or not. In such a case that the IoT device is not satisfied with the fog results and decides not to pay, it may start a dispute with a third party to retrieve its deposit.

Note, that this proposed system [18] requires a separate blockchain transaction for the 1) initial deposit, 2) a confirmation of deposit from fog, 3) sending a separate payment from IoT to fog, 4) then returning the deposit to IoT. This payment process can be costly in blockchain fees due to the total number of required transactions. Furthermore, this process requires verification of the result from the IoT device, which may not be computationally possible from resource-constrained devices.

D. Blockchain-fog integration

Fog nodes are geographicaly distributed, and blockchains are replicated and hosted on distributed servers. Therefore, it is reasonable to combine these concepts to minimize the communication latency between fog nodes and the blockchain. *Blockchain-enabled fog nodes* are fog nodes that use a portion of their resources to host a copy of the blockchain. By doing so, all communication delay between fog and blockchain is eliminated. Almadhoun et al. [23] uses blockchain-enabled fog nodes for IoT authentication – a process whose speed is highly dependent on communication delay between IoT, fog and blockchain. The resource requirements of the blockchain can be further reduced by using 'light nodes' which use block summarization to reduce the amount of data stored on the fog node [19], [30], [31]. In particular, in this paper we store only blockchain data relevant to the authentication and auditing processes.

E. Summary of Reviewed Literature

None of the reviewed contributions consider a penalty or action to be taken if a fog node or edge server is found to be corrupted. If the corruption is accidental, then the appropriate server could be given the correct data. However, if the corrupted data is malicious, then there is no penalty to stop the server from continuing to alter cached or processed IoT data. At worst, a fog node that returns malicious data is simply not paid [18].

In public systems, a call-and-response process of requesting a proof of data integrity from servers must be taken. However, there is no incentive given for the data servers to comply with the audit [12], [15], [27]. To validate service integrity, it is left to the IoT device to verify the work done by the fog is correct before giving payment, a task which is not always computationally feasible by resource constrained IoT [18]. Even in private networks [28], it may not be reasonable to have a third party auditor with full accessibility of client files without major privacy concerns of individuals.

The focus of this paper is to validate the service integrity of fog nodes who are meant to support IoT. We take inspiration from related work to form the FISIE system, a blockchain-based system that streamlines IoT-fog mutual authentication, fog service monetization, and verification of fog integrity. We also recognize the benefits of integrating blockchain within the fog layer for low-latency mutual authentication. We find the addition of an incentive and penalty mechanism to be necessary to enforce auditing cooperation from fog nodes and overall IoT-fog system integrity. A comparison of the proposed FISIE system with other contributions is shown in Table I.

III. FISIE System Overview

The FISIE system aims to 1) streamline IoT-fog mutual authentication and fog service monetization, 2) verify the integrity of fog nodes via external service auditing, and 3) promote the honest collaboration between IoT and fog via incentives and penalties. These objectives are accomplished via the *Identity & Integrity Management Smart Contract* (IIMSC) which interacts with IoT, fog, and an *oracle* – an off-chain semi-trusted third-party. A generalized blockchain structure will increase the likelihood by others of adopting the FISIE system. Hence, though our default implementation uses Ethereum [20], other implementations may use smart contract-capable blockchain platforms such as Solana[1] or Layer 2 platforms such as Arbitrum[2] or Optimism[3] for scalability and lower processing fees [32], [33].

The key processes of IIMSC are summarized as 1) Identity Management, 2) Payment Management, and 3) Integrity Verification. The key processes of IIMSC are shown in Fig. 1. These key components are summarized below, and further explored in sections VI, VII, and VIII. These three components offer incentives and penalties for fog nodes to behave honestly (see Section IX).

A. Identity Management

IIMSC defines *lookup tables* that hold information about IoT and fog blockchain addresses, current token holdings, and fog reputation. These lookup tables are used for mutual

[1]https://solana.com/
[2]https://arbitrum.io/
[3]https://www.optimism.io/
[4]IoT icons by https://www.avsystem.com

Contribution	Mutual Authentication	Fog Service Monetization	Resource Constrained IoT-Compatible	Immutable (Blockchain)	Fog Penalization	Fog Honesty Verification	Public Fog Auditing	Penalization for Malicious Fog
Debe [17]	✗	✓	✓	✓	✗	✗	✗	✗
Almadhoun [23]	✓	✗	✓	✓	✗	✗	✗	✗
Singh [25]	✓	✗	✓	✗	✗	✗	✗	✗
Patwary [26]	✓	✗	✓	✓	✗	✗	✗	✗
Tian [29]	✗	✗	✗	✗	✗	✓	✓	✗
Huang [18]	✗	✓	✗	✓	✗	✓	✗	✗
FISIE	✓	✓	✓	✓	✓	✓	✓	✓

Table I: Inspiration for the FISIE system is taken from various contributions.

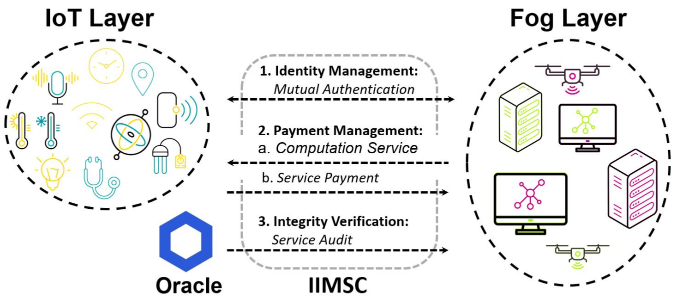


Figure 1: The main objectives of the FISIE system is to use a blockchain smart contract to 1) facilitate authentication and service and payment between IoT devices[4]and fog nodes, and 2) enforce service integrity among fog nodes.

authentication between IoT devices and fog nodes. Prior to participating in the FISIE system, all IoT devices and fog nodes must register with IIMSC. The blockchain addresses are used both to validate an entity's identity, as well as to forward payment. Current IoT token holdings are listed to ensure IoT devices have enough funds to pay for fog services. Fog devices have two fields for current token holdings – one for a collateral deposit to use the FISIE system, and another to accrue fog service payments. Both collateral deposit and a fog's reputation score give indication of a fog node's past behaviour, whether honest or malicious, i.e., their *reliability*. Hence, IoT devices may filter candidate fog nodes to which it is comfortable sending data based on a fog's reliability.

B. Payment Management

Once an IoT-fog pair has authenticated each other, an IoT device may send its request to the fog node for service. This is done off-chain via ECC encryption [34]. Once service is complete and successfully returned to the IoT device, a predetermined amount is transferred from the IoT device's funds to the fog node's service payment funds via IIMSC lookup tables. That is, all funds remain within IIMSC until the fog node withdraws them.

C. Integrity Verification

To the best of our knowledge, there does not exist any research contributions in the enforcement of fog service integrity. We seek to implement a fog service auditing mechanism based on a call-and-response for the service output from fog nodes for an IoT request. Unlike previous contributions, we include both an incentive to encourage fog nodes to participate in the call-and-response, and a penalty in the case the fog node fails the service audit.

We assume that IoT devices have limited resources and are therefore unable to verify the correctness of fog node's service. Therefore, we define a service auditing scheme to allow an oracle to verify the integrity of fog nodes without revealing its identity. Indeed, if a fog node was aware that it was being audited by the oracle, it may change it's behaviour. Instead, by using a Zero-Knowledge Proof of Membership [35], the oracle disguises itself as another IoT device, encouraging the fog node to behave as it normally does.

D. Penalty & Incentive Mechanisms

The existence of an auditor in itself acts as a deterrent to malicious behaviour from fog nodes. If a fog node is found to return a faulty response, we employ a *penalty* mechanism to both reduce the fog node's collateral deposit and reputation score. If the audit is successful, the fog node's reputation score may increase up to a fixed cap. A higher reputation score may lead to more service requests from IoT, and hence more service payments. Therefore, such a reward also acts as an *incentive* for fog nodes to behave honestly.

IV. Domain Background & Configurations

In this section, we provide background knowledge of blockchain, the IoT-fog environment, and cryptography, and define their role in the FISIE system.

A. Blockchain

Public blockchains, such as Bitcoin and Ethereum, are decentralized ledgers that enforce block consensus across all immutable blockchain nodes, which allows them to operate in a trustless environment [36]. Public blockchains also provide pseudo-anonymity, fault tolerance, and auditability. Blockchain technology is viewed as a key technology in adding security & privacy to IoT and industrial IoT (IIoT) applications [37].

1) Smart Contracts: A *smart contract* is an agreement between two or more parties that self-executes when specific conditions are met [38]. A smart contract on the blockchain benefits from the same immutability, persistency and auditability as other blockchain transactions. A common use case for blockchain smart contracts is providing access control of IoT data, which can mitigate security & privacy issues in IoT [39]. The FISIE system's implementation of the *Identity & Integrity Management Smart Contract* (IIMSC) is compatible with any smart contract-capable blockchain.

2) Oracles: Often, a smart contract is dependent on real-world data to determine when its execution conditions are satisfied. However, the blockchain is isolated from the real-world internet environment, creating a need for a separate entity to convey the appropriate external information to the blockchain. An *oracle* is an off-chain third-party that is used to inject external data into smart contracts. Since oracles operate off-chain, it is necessary to validate the trustworthiness of both the oracle and the external data sources [40]. For this reason, centralized oracles are not often used since the validity of the communicated data from a single centralized entity cannot be trusted. Instead, multiple decentralized oracles are often used to cross-verify each other and create a trusted data feed [41]. The FISIE system relies on external decentralized oracles to audit fog nodes and trigger the appropriate smart contracts.

B. Elliptic Curve Cryptography – Definitions & Settings

The FISIE system uses *elliptic curve cryptography* (ECC), a public-key cryptographic method that uses a globally agreed upon elliptic curve and base point over a finite field to generate public and private keys [34].

Consider the finite field $\mathbb{F}_p$ for large prime number p. Over the elliptic curve E, beginning from a base point G, we select a secret key k and derive the public key P as

$$P = k \cdot G \tag{1}$$

where we 'add' $(\cdot)$ G k times over the finite field of E.

1) The elliptic curve: Bitcoin and Ethereum use the `secp256k1` system for the Elliptic Curve Digital Signature Algorithm (ECDSA) [42] to sign blockchain transactions. The `secp256k1` elliptic curve is defined as

$$E : y^2 = x^3 + 7, \tag{2}$$

and produces 256-bit keys [43], [44]. Unlike other contributions that trust a third party to define the elliptic curve parameters and generate device keys [14], [25], we will simplify the process and use the built-in ECC parameters of these blockchains for secure IoT-fog communication.

2) Encryption: Suppose entity A has public key P_A and private key k_A, and entity B has public key P_B and private key k_B. Then a symmetric key can be formed between A and B since,

$$k_B P_A = k_B \cdot (k_A G) = k_A \cdot (k_B G) = k_A P_B. \tag{3}$$

Using this symmetric key, we can encrypt and decrypt a message between A and B using a symmetric encryption algorithm such as AES [34], [45].

3) ECDSA: ECDSA is the primary signature generation and verification algorithm in Bitcoin and Ethereum blockchains [44]. No transaction is accepted by the blockchain without a valid signature. Suppose a user A submits a signature to a verifier B. ECDSA enables a verifier B to recover the public key P_A from a valid signature s. Hence, there is no need for A to submit P_A to B. For the remainder of this paper, every signature used is an ECDSA signature.

4) One-way hash function: We define the function $H : \{0,1\}^* \mapsto \{0,1\}^{256}$ as a secure, one-way 256-bit *hash* function. Examples of viable hash functions with this property are `SHA-256` and `Keccak-256` used by Bitcoin and Ethereum ECDSA respectively [36], [20]. Importantly, these hash functions also derive a user's address by hashing the user's public key. The address of a user with public key P is defined as the last 20 bytes of the hash $H(P)$ [20]. We define the operation $||_n$ to be the n-byte right hand truncation of a value. Hence, a user with public key P has address $H(P)||_{20}$.

C. Physical Architecture

Our proposed architecture is meant to be as general as possible so it may fit any existing IoT-fog infrastructure. Both IoT devices and fog nodes are heterogeneous and distributed. The architecture is divided into an IoT layer, a fog layer, and a Cloud layer. For simplicity of discussion, we consider the blockchain as part of the fog layer.

1) IoT layer: The IoT layer is composed of devices with varying resource capabilities and requirements from higher layers. We focus our approach on devices with limited computing/storage resources that require processing from the fog layer, and may send data to the Cloud layer for long-term storage. We define I as the set of IoT devices in the FISIE system. All IoT devices in I have at least enough resources to store the necessary encryption keys and to communicate with higher layers.

2) Fog layer: The fog layer is composed of fog nodes, oracles, and blockchain nodes for a smart contract-capable blockchain. We define F as the set of fog nodes and O as the set of oracles in the FISIE system. The fog nodes in F have varying resource capabilities, and some may have sufficient resources to run a light blockchain node [30], a blockchain oracle [41], or both. Blockchain nodes may exist separately, or within a fog node, i.e., blockchain-enabled fog nodes [19].

3) Cloud layer: The Cloud layer is composed of mega data-centers capable of long-term data storage and substantial computing power [46]. Data that require storage may come from the fog layer after it has been processed, or directly from the IoT layer.

V. IIMSC - Initialization

Beginning in this section, and continuing in sections VI, VII and VIII, we define in detail the functionalities of IIMSC, including its initial parameters and tables. Every function of IIMSC takes a signature s as a final argument, which is validated via ECDSA before executing the function. Therefore, we omit the signature validation from the description of IIMSC functions. The functions and lookup tables of IIMSC are designed to 1) facilitate the mutual authentication process, 2) provide security and accountability to the IoT-fog service payment process, and 3) enable incentive and penalty mechanisms for fog integrity. A summary of all IIMSC functions are provided in Table II and are described in future sections.

Table II: A summary of IIMSC functions

Initialisation	`Initialisation()`
Registration	`IoT_registration(Ether` E_u`)`
	`Fog_registration(Ether` E_u`)`
	`Oracle_registration()`
Funds	`IoT_add_funds(Ether` E_u`)`
	`IoT_withdraw_funds(float` u`)`
	`Fog_withdraw_funds(float` u`)`
Removal	`IoT_remove()`
	`Fog_remove()`
Payment	`IoT_fog_payment(float` d`)`
Audit result	`Fog_reward(Address` a_f`, Ring sign.` $\mathcal{R}_\omega$`)`
	`Fog_penalize(Address` a_f`, Ring sign.` $\mathcal{R}_\omega$`)`

A. Lookup tables

Secure mutual authentication schemes rely on a trusted third-party to validate the identity of each authenticating member [23], [25]. Therefore, we propose that IoT devices and fog nodes register with respective IoT and fog lookup tables on the blockchain. The registration process uses ECDSA signatures to initially confirm the identity of the registering party. Hence, all registration information on the lookup tables are publicly accessible and pre-verified. To register with the blockchain, we require a payment deposit from IoT devices, and a collateral deposit from fog nodes. These deposited amounts are reflected in the lookup tables. Since our default implementation uses Ethereum, all mentions of payments, funds and deposits will use the Ether cryptocurrency [20].

For each IoT device $i \in I$, the fields in the IoT lookup table T_I are defined as

- **IoTAddress**: The address of the IoT device i, which is also the truncated hash of the IoT public key $H(P_i)||_{20}$.
- **AvailFunds**: The available funds of IoT device i. These funds are used to pay for fog services.

For record $\mathbf{t}_i \in T_I$ of IoT device i, we denote these entries as $\mathbf{t}_i.A$, and $\mathbf{t}_f.AF$ respectively.

For each fog node $f \in F$, the fields in the fog lookup table T_F are defined as

- **FogAddress**: The address of the fog node f, which is also the truncated hash of the IoT public key $H(P_f)||_{20}$.
- **Deposit**: The collateral deposit given by fog node f. A portion of the deposit may be lost as a penalty for failing a service audit.
- **AvailFunds**: The available funds of fog node f. Available funds come from IoT service payments and may be withdrawn at the fog node's discretion.
- **Reputation**: The reputation score of fog node f. This reputation score is updated based on the results of a service audit. IoT devices may choose which fog nodes to work with based on their respective reputation scores.

For record $\mathbf{t}_f \in T_F$ of fog node f, we denote these entries as $\mathbf{t}_f.A$, $\mathbf{t}_f.AF$, $\mathbf{t}_f.D$, and $\mathbf{t}_f.R$ respectively.

By default, the lookup tables are implemented on-chain. Alternatively, the tables may be placed off-chain and managed by a *reverse oracle*, i.e., an outbound oracle that executes on behalf of the blockchain [47]. In this case, we add an additional column to both tables labeled 'LastUpdateHeader'. For each table record, the 'LastUpdateHeader' field contains the blockchain header associated with the latest record update. By referencing this hash in the lookup tables, we also enforce immutability on the values of the off-chain lookup table.

In addition, we define an on-chain oracle lookup table T_O to register any oracle that wishes to participate in the service auditing process. T_O has a single field **OracleAddress**, denoted $\mathbf{t}_o.A$ for record $\mathbf{t}_o \in T_O$ of oracle $o \in O$.

B. Initialization

The `Initialization` function is the constructor of IIMSC. It creates the IoT, fog and oracle lookup tables $\{\text{IIMSC}.T_I, \text{IIMSC}.T_F, \text{IIMSC}.T_O\}$, and sets the following parameters:

- the minimum, initial and maximum reputation scores $\{\text{IIMSC}.R_{\texttt{Min}}, \text{IIMSC}.R_{\texttt{Init}}, \text{IIMSC}.R_{\texttt{Max}}\}$, where $\text{IIMSC}.R_{\texttt{Min}} \leq \text{IIMSC}.R_{\texttt{Init}} \leq \text{IIMSC}.R_{\texttt{Max}}$
- the reputation penalty and reward $\{\text{IIMSC}.r^-, \text{IIMSC}.r^+\}$, where $\text{IIMSC}.r^- > \text{IIMSC}.r^+$
- the fog collateral deposit amount $\text{IIMSC}.D$ and penalty deposit deduction $\text{IIMSC}.d^-$

It is important that the reward r^+ is smaller than the penalty r^- to deter fog nodes from behaving outside of what is expected.

C. IIMSC pooled funds

During registration process, IoT devices, fog nodes and the oracles each submit deposits, either for payments or as collateral. These funds are 'moved' during the payment and penalty processes. All funds deposited into IIMSC are pooled within the smart contract, and the individual token holdings are detailed in the lookup table for each device. Hence, any payments that occur through IIMSC have no actual transfer of payments between devices. Rather, the lookup table values are updated, and token changes are realized upon withdrawal.

VI. IIMSC – Identity Management

The objective of the identity management functions of IIMSC is to facilitate mutual authentication between IoT and fog. Prior to sending a request, the IoT device must authenticate a fog node by verifying it is registered in T_F and has a sufficient reputation score. Likewise, the fog node must authenticate the IoT device to ensure it is registered in T_I and has sufficient funds to pay the fog node.

A. Registration

Once IIMSC has initialized, any entity that wishes to partake in the FISIE system must first register with the blockchain.

1) IoT registration: The `IoT_registration` function takes an itial Ether deposit E_u of amount $u > 0$ from IoT device $i \in I$. After ensuring $a_i = H(P_i)||_{20}$ is not already in T_I, the values $\mathbf{t}_i.A \leftarrow a_i$ and $\mathbf{t}_i.AF \leftarrow u$ are added to new record $\mathbf{t}_i \in T_I$.

2) Fog registration: The `Fog_registration` function takes a deposit E_d of amount d for fog node $f \in F$, with $d \geq$ IIMSC.D. An amount IIMSC.D is used for the deposit, and the remainder $v = d -$ IIMSC.D is set as the initial available funds. The reputation of f is set to the initial reputation $r =$ IIMSC.$R_{\texttt{Init}}$. After ensuring $a_f = H(P_f)||_{20}$ is not already in T_F, the values $\mathbf{t}_f.A \leftarrow a_f$, $\mathbf{t}_f.D \leftarrow$ IIMSC.D, $\mathbf{t}_f.AF \leftarrow v$ and $\mathbf{t}_f.R \leftarrow r$ are added to the new record $\mathbf{t}_f \in T_F$.

3) Oracle registration: An oracle $o \in O$ with public key P_Ω must register with the blockchain in order to securely communicate its oracle results. Therefore, we define a function `Oracle_registration` to create a record $\mathbf{t}_o \in T_O$ with $\mathbf{t}_o.A \leftarrow H(P_\Omega)||_{20}$.

B. Removal

If an IoT device $i \in I$ no longer wishes to be a part of the network, it may call the `IoT_remove` function. All available funds under the record $H(P_i)||_{20}$ are returned to IoT device i, and the record is removed from T_I.

A fog node may request to exit the system, or it may be removed forcefully by IIMSC. In both cases, the `Fog_remove` function is executed. If the `Fog_remove` function is initiated by the fog node, then the remaining deposit and available funds under the record $H(P_f)||_{20}$ is returned to fog node f, and the record is removed from T_F. If at any point the fog deposit reaches zero or the reputation score falls below IIMSC.$R_{\texttt{Min}}$, the `Fog_remove` function is automatically triggered, the remaining available funds and deposit (if any) are returned and the fog node is removed from T_F.

C. Mutual Authentication

We outline the mutual authentication process between an IoT device $i \in I$ and a fog node $f \in F$.

1) IoT device i sends a signature s_i to fog node f.
2) Fog node f will recover P_i from s_i, and query $H(P_i)||_{20}$ as the address of i from the IoT lookup table T_I.
3) If the IoT address is found in step 2), continue with step 4). If not, mutual authentication fails.
4) Fog node f sends a signature s_f to IoT device i.
5) IoT device i will simultaneously
   (a) recover P_f from s_f and query $H(P_f)||_{20}$ as the address of f from the IoT lookup table T_F.
   (b) verify that the reputation score $\mathbf{t}_f.R$ meets the reputation threshold $i.R$ set by i.
6) If the fog address is found and the reputation threshold is met in step 5), continue to step 7). If not, mutual authentication fails.
7) IoT device i and fog node f have successfully completed mutual authentication, and established a symmetric key $P_f k_i = P_i k_f$ for secured communication. They may begin to collaborate.

The mutual authentication process is summarized in Fig. 2.

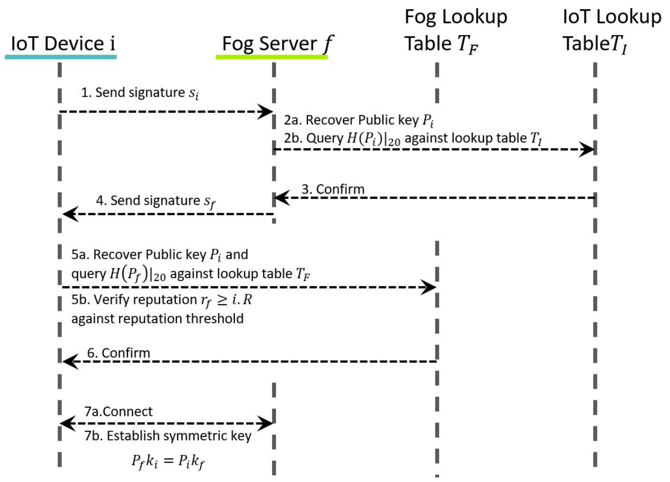

IoT Device i
Fog Server f
Fog Lookup Table T_F
IoT Lookup TableT_I
1. Send signature s_i
2a. Recover Public key P_i
2b. Query $H(P_i)|_{20}$ against lookup table T_I
3. Confirm
4. Send signature s_f
5a. Recover Public key P_i and query $H(P_f)|_{20}$ against lookup table T_F
5b. Verify reputation $r_f \geq i.R$ against reputation threshold
6. Confirm
7a.Connect
7b. Establish symmetric key $P_f k_i = P_i k_f$

Figure 2: The mutual authentication process. Recall, every function takes a signature, from which the associated public key is recovered.

VII. IIMSC – Payment Management

The objective of the payment management functions of IIMSC is to streamline the IoT-fog service payment process, while also providing security and accountability. Since all payment transactions are posted on the blockchain, visibility of payment records can be used in the case of a payment dispute. Once an IoT device and fog node are mutually authenticated, the IoT device may send any computation requests to the fog node in exchange for a portion of deposited funds.

A. Addition and withdrawal of funds

Once a device has registered with IIMSC, it may add or withdraw funds used for service payments.

1) IoT funds: After an IoT device i has registered with the blockchain, it may add additional funds to its available reserve. The `IoT_add_funds` function takes additional funds E_u of amount $u > 0$. The record $\mathbf{t}_i \in T_I$ where $\mathbf{t}_i.A = H(P_i)||_{20}$ is updated with $\mathbf{t}_i.AF \leftarrow \mathbf{t}_i.AF + u$. Similarly, the `IoT_withdraw_funds` may be used to withdraw an amount $u \in (0, \mathbf{t}_i.AF]$ from the available funds. Then, the record $\mathbf{t}_i \in T_I$ is updated with $\mathbf{t}_i.AF \leftarrow \mathbf{t}_i.AF - u$ and E_u Ether is sent to IoT device i.

2) Fog funds: Once fog node f begins to service IoT requests, it will accumulate payments in its available funds. Fog node f may request to withdraw an amount $u \leq \mathbf{t}_f.AF$ through the `Fog_withdraw_funds` function. The `Fog_withdraw_funds` function takes an amount to withdraw $u \in (0, \mathbf{t}_f.AF]$. The record $\mathbf{t}_f \in T_I$ where $\mathbf{t}_f.A = H(P_f)||_{20}$ is updated with $\mathbf{t}_i.AF \leftarrow \mathbf{t}_i.AF - u$, and E_u Ether is sent to f.

B. IoT-Fog service and payment

Once the mutual authentication process is successfully completed and a symmetric key $P_i k_f = P_f k_i$ has been established between $i \in I$ and $f \in F$, IoT device i may request computational support from fog node f via symmetric encryption. We outline the IoT-fog service process between IoT device i and fog node f.

1) IoT device i transmits a proposed payment d, a package g, and the signature s_i of the transaction to fog node f.
2) If f doesn't accept, it sends a 'reject' return statement OR lets the request time out. The process ends.
3) Else, f processes package g and gets result τ.
4) The fog node f returns result τ to IoT device i.
5) IoT device i triggers the `IoT_fog_payment` function with parameters: agreed payment d and IoT signature s_i.
6) The `IoT_fog_payment` function verifies signature s_i, and transfers an amount d from $\mathbf{t}_i.AF$ to $\mathbf{t}_f.AF$, $\mathbf{t}_i \in T_I$, $\mathbf{t}_f \in T_F$.

The service and payment process is summarized in Fig. 3.

C. Matching, Bargaining, and Disputes

The default implementations of mutual authentication and service payment described above are streamlined, without consideration of fog selection, price bargaining or payment disputes. In reality, there is room for flexibility to address these concerns in these processes.

Prior to mutual authentication, IoT devices must select to which fog node it wishes to contact for service. There exist many, more sophisticated matching algorithms for pairing IoT devices with fog nodes based on proximity and available fog resource capacity [48].

Once devices have been authenticated and the IoT device submits its request with proposed payment, the fog node may counter-offer. That is, the IoT device and fog node may enter into a round of bargaining to determine an agreed price [49]. Indeed, the matching and bargaining processes can even be combined into an auction-based process whereby fog nodes are matched by bidding on the IoT request [50].

Once the service process is complete and the fog node has returned the processed result τ of package g, it is up to the IoT device to trigger the `IoT_fog_payment` function. If it does not within a reasonable amount of time, the fog node may start a dispute with a decentralized dispute resolution platform such as Kleros[5] [51]. Conversely, if the IoT device does not recieve a response from the fog in a timely manner, the IoT device may start a dispute. These dispute resolution processes may prove to be costly for IoT devices and fog nodes, thus incentivizing timely processing of IoT requests, and timely triggering of the payment smart contract.

VIII. IIMSC – Integrity Verification

By occasionally checking the processing results of fog nodes, we can add a level of integrity and trust to the IoT-fog environment. We rely on trusted decentralized oracles to audit and verify the integrity of fog nodes. An oracle $o \in O$ uses two key pairs – one registered as an 'IoT device' and one registered as an oracle. All audits are submitted by the address associated to the IoT lookup table T_I, so that fog nodes believe the audit is a normal IoT request. This is crucial to verify the natural behavior of a fog node when not under supervision.

[5]https://kleros.io/

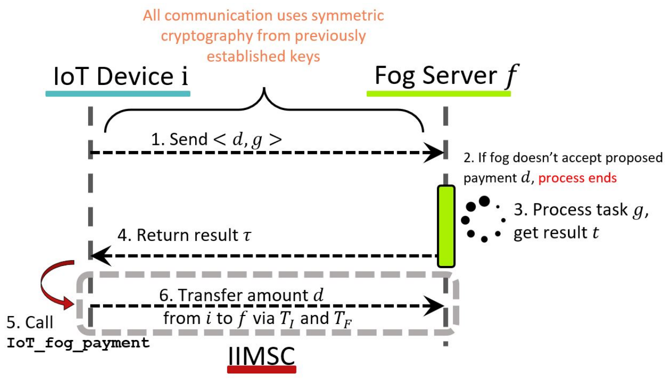

All communication uses symmetric cryptography from previously established keys
IoT Device i
Fog Server f
1. Send $< d, g >$
2. If fog doesn't accept proposed payment d, process ends
3. Process task g, get result t
4. Return result τ
5. Call IoT_fog_payment
6. Transfer amount d from i to f via T_I and T_F
IIMSC

Figure 3: The IoT-fog processing and payment workflow.

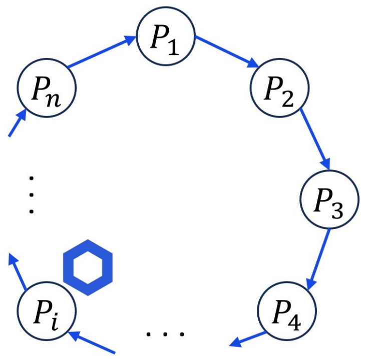


Figure 4: A ring signature, zero-knowledge proof of membership, hides the identity of the oracle among IoT.

A. Ring Signature - Proof of Membership

Suppose oracle $o \in O$ has two key pairs (P_ω, k_ω) and (P_Ω, k_Ω). Prior to auditing, oracle o registers itself with the blockchain as an IoT device using key pair (P_ω, k_ω), and as an oracle using key pair (P_Ω, k_Ω). These key pairs result in different hashes across different tables. Therefore, the identity of oracle o on the IoT table is not known to fog nodes nor the blockchain. Hence, when submitting an audit result to the blockchain, the oracle must prove that the audit came from a valid IoT service request. In other words, it must prove that it belongs to the IoT lookup table without revealing the identity of its IoT address.

This problem relates to the class of zero knowledge proofs for set membership [35], which have been proven essential in blockchain applications [52]. Given the information publicly available in the IoT lookup table, we choose to implement a *ring signature scheme* [53].

Let P_I be the set of public keys from table T_I. We choose some enumerated subset $\bar{P} \subseteq P_I$ containing P_ω such that $|\bar{P}| = n$ and $\bar{P} = \{P_1, P_2, \ldots, P_n\}$ where $P_\omega = P_j$ for some $1 \leq j \leq n$.

An oracle with key-pair $(P_\omega, k_\omega) = (P_j, k_j)$ builds a ring signature of message m over the elliptic curve of prime p and base point G as follows:

1) Choose a random integer $q \in [0, p-1]$.
2) Calculate $T_j = (x_j, y_j) = q \cdot G$.
3) $\forall i = 1, \ldots, n$, $i \neq j$, pick random integers $\sigma_i \in [0, p-1]$.

4) for $i = j+1, \ldots, n, 1, \ldots, j-1$:
   $c_i = H(m|x_{i-1})$
   $T_i = (x_i, y_i) = \sigma_i \cdot G + c_i \cdot P_i$
5) $c_j = H(m|x_{j-1})$
6) $\sigma_j = q - c_j k_j$.

Let $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_n]$ and $\boldsymbol{P} = [P_1, \ldots, P_n]$. Oracle o submits $(c_1, \boldsymbol{\sigma}, \boldsymbol{P})$.

1) Verification: The verifier (IIMSC) begins with c_1, and caluclates

$T_1 = (x_1, y_1) = \sigma_1 \cdot G + c_1 \cdot P_1$
for $i = 2, \ldots, n$:
   $c_i = H(m|x_{i-1})$
   $T_i = (x_i, y_i) = \sigma_i \cdot G + c_i \cdot P_i$
$c_1' = H(m|x_n)$.

The ring signature is accepted if $c_1 = c_1'$.

2) Correctness: : By the original choice of $T_j = q \cdot G$, the final choice of s_j 'closes' the ring. That is,

$$\begin{aligned} T_j &= \sigma_j \cdot G + c_j P_j \\ &= (q - c_j k_j) \cdot G + c_j k_j \cdot G \\ &= q \cdot G. \end{aligned}$$

B. Reward & Penalty Functions

IIMSC alters the deposit and reputation scores of fog nodes based on the results of a service audit. Both the `Fog_reward` and `Fog_penalize` functions take a ring signature $\mathcal{R}_\omega$ from an oracle $o \in O$. If the ring signature is valid, then there exists an address in T_I that belongs to oracle o.

1) Passed audit: When a fog node $f \in F$ passes a service audit sent by oracle $o \in O$ with public key P_Ω, the oracle calls the `Fog_reward` function which takes the fog address a_f and a ring signature $\mathcal{R}_\omega$. After verifying $H(P_\Omega)||_{20}$ is in T_O, a_f is in T_F and verifying the validity of ring signature $\mathcal{R}_\omega$, the function increases the fog reputation by an amount IIMSC.r^+, up to a maximum IIMSC.$R_{\texttt{Max}}$. That is, $\mathbf{t}_f.R \leftarrow \min\{\mathbf{t}_f.R + \text{IIMSC}.r^+, \text{IIMSC}.R_{\texttt{Max}}\}$.

2) Failed audit: When a fog node $f \in F$ fails a service audit sent by oracle $o \in O$ with public key P_Ω, the oracle calls the `Fog_penalize` function which takes the fog address a_f and a ring signature $\mathcal{R}_\omega$. After verifying $H(P_\Omega)||_{20}$ is in T_O, a_f is in T_F, and verifying the validity of ring signature $\mathcal{R}_\omega$, the function 1) decreases the fog reputation by an amount IIMSC.r^-, and 2) decreases the deposit by an amount IIMSC.d^-, or to 0, whichever is higher. That is, $\mathbf{t}_f.R \leftarrow \mathbf{t}_f.R - \text{IIMSC}.r^-$ and $\mathbf{t}_f.D \leftarrow \max\{\mathbf{t}_f.D - \text{IIMSC}.d^-, 0\}$. The lost deposit is distributed among the registered IoT devices. If the updated reputation $\mathbf{t}.R$ falls below IIMSC.$R_{\texttt{Min}}$, or if the updated deposit $\mathbf{t}.D$ reaches 0, then `Fog_remove` is automatically called on f.

C. Service Audit

We define the IoT address $a_\omega = H(P_\omega)||_{20}$ and oracle address $a_\Omega = H(P_\Omega)||_{20}$ as the two addresses used by oracle $o \in O$ for fog and blockchain communication respectively.

1) Oracle o and fog node f mutually authenticate and establish a symmetric key $P_\omega k_f = P_f k_\omega$.
2) Oracle o sends a package g to f following the process in section VII-B.
3) Simultaneously, oracle o
   a) waits for request response τ_f from f.
   b) calculates the expected output τ_ω of g.
4) Oracle o computes a ring signature $\mathcal{R}_\omega = \{c_1, \boldsymbol{\sigma}, \boldsymbol{P}\}$ and compares the fog result τ_f with the expected result τ_ω.
   a) If $\tau_f = \tau_\Omega$, fog node f has passed the service audit. Oracle o calls the `Fog_reward` function with fog address a_f, oracle signature s_Ω, and ring signature $\mathcal{R}_\omega$.
   b) Else, if $\tau_f \neq \tau_\Omega$, and f has failed the service audit. Oracle o calls the `Fog_penalize` function with fog address a_f, oracle signature s_Ω, and ring signature $\mathcal{R}_\omega$.

1) Oracle payment: When an oracle o executes a service audit, it takes time and uses processing resources for the benefit of the FISIE system. In addition, since oracle o is disguising a service audit as an IoT service request, it must pay a service fee to the audited fog node. In both cases, the oracle should be fairly compensated and reimbursed for its efforts.

By default, IIMSC takes a service fee from IoT devices whenever a call to `IoT_fog_payment(`d`)` is made. That is, IIMSC takes a small portion of the service payment d as the service fee. These fees are pooled by IIMSC. A portion of the pool is used to pay the oracles, and the rest is used to pay the owners of the smart contract.

2) Scheduling policy: The audit scheduling policy defines how often oracles can execute service audits. By default, one oracle completes one service audit every η requests, where η is defined by IIMSC. A large η ensures that more than enough fees have been collected to pay the oracle fairly, but may not result in frequent enough service audits. In contrast, a small η results in more, frequent service audits, but would require larger fees to be taken from IoT service payments to cover the oracle costs. Other more sophisticated scheduling policies [54], [55] can be considered for service auditing that take into account the overall health of the system. This is left for future work.

IX. IIMSC – Penalty & Incentive Mechanisms

The lookup tables, IIMSC parameters, and integrity verification functions are used to provide incentives and penalties for fog nodes to encourage integrity.

A. Fog monetization

An IoT device $i \in I$ may request service from a fog node, in exchance for a proposed payment d, where $d \leq \mathbf{t}_i.AF$. That is, the IoT device has sufficient available funds to satisfy the proposed payment. Once a fog node has serviced an IoT request, the IoT device pays the fog node for its services. This IoT payment provides a **monetary incentive** to fog nodes to service IoT. A service payment from an IoT device is deducted from its available funds in T_I, which is the total of all previously deposited and unspent funds, in IIMSC, from the IoT during or after registration.

B. Fog collateral deposit

Upon registry, a fog node $f \in F$ has collateral deposit $\mathbf{t}_f.D = \text{IIMSC}.D$. Periodically, a service audit is sent out to fog node f by an oracle posing as an IoT device. The fog node, unaware the request is from an oracle, would respond to the request normally, either with a correct or faulty response. If the response is incorrect, i.e., fog node f has failed the service audit, then a portion of the fog deposited funds $\mathbf{t}_f.D$ are deducted from T_F and redistributed to IoT. This loss of collateral provides a **monetary penalty** to fog nodes if they fail a service audit. If a fog node loses its entire deposit, i.e., $\mathbf{t}_f.D = 0$, then the fog node is removed from T_F, and hence, from the FISIE system.

By default, the collateral deposit amount IIMSC.D is fixed, and any additional deposit is converted to available funds. Alternatively, a possible implementation of IIMSC could allow for flexible deposit amounts, and a more sophisticated deposit deduction or reduction mechanisms [56]. For example, IIMSC could decrease the required deposit from long-term behaving fog nodes. In this case, the additional deposit over the newly reduced deposit threshold is converted to available funds that the fog node may withdraw. This implementation provides an additional incentive to fog nodes to behave properly over the long-term.

C. Fog reputation

Fog nodes are given a reputation score in lookup table T_F. The reputation score is updated by IIMSC based on the results of a service audit. That is, the reputation in T_F is decreased if a fog node fails a service audit, and is increased if it passes. An IoT device may filter the fog nodes based on their reputation score before choosing where to send a request. Therefore, it is beneficial to the fog node to always behave properly, as to have a higher reputation and the possibility for more IoT requests, i.e., IoT payments. This provides a **service incentive** to fog nodes if they pass a service audit. Conversely, a loss of reputation provides a **service penalty** to fog nodes if they fail a service audit. Every fog node begins at the same initial reputation score IIMSC.$R_{\texttt{Init}}$, can increase up to a fixed maximum score IIMSC.$R_{\texttt{Max}}$, and is removed from the system if the score falls below a minimal reputation threshold IIMSC.$R_{\texttt{Min}}$.

By default, IIMSC will decrease and increase the reputation score by a fixed IIMSC.r^- and IIMSC.r^+ respectively, where IIMSC.$r^- >$ IIMSC.r^+. However, more sophisticated strategies for calculating [57] and updating [58] reputation score can be used, taking into account factors such as the number of IoT requests, and service level agreement (SLA) compliance [59].

D. Fog deposit distribution

If a fog node fails an audit, an amount d^- is deducted from its collateral deposit on the fog lookup table T_F. The associated Ether E_{d^-} is still attached to IIMSC. Hence, we choose to distribute the amount IIMSC.d^- amount among a subset of IoT devices $\bar{I} \subseteq I$ by updating available funds in the lookup table T_I by an equivalent amount. That is, for each $i \in \bar{I}$, we select an amount $d_i > 0$ such that $\sum_{i \in \bar{I}} d_i = \text{IIMSC}.d^-$, and increase the available funds $\mathbf{t}_i.AF \leftarrow \mathbf{t}_i.AF + d_i, \forall i \in \bar{I}$.

By default, IIMSC distributes the deducted deposit from $f \in F$ equally among all IoT devices by an amount d^-/n where $n = |T_I|$. Other distributions strategies are possible, such as distributing only to IoT devices that have previously been serviced by f in either an equal or weighted manner.

X. Simulation of Integrity

To the best of our knowledge, no other contribution provides both an incentive and a penalty mechanism to enforce the integrity of an IoT-fog system. We first discuss the security of the FISIE system. To determine the effectiveness of our proposed system, we execute an auditing simulation over a set of malicious nodes. We also define several auditing scheduling policies to compare their effectiveness amongst each other.

A. Security Analysis – Discussion

The FISIE system streamlines the IoT-fog mutual authentication, service and payment processes while maintaining secure and accountable IoT-fog communication.

1) Encrypted Communication: All direct communication between IoT and fog is encrypted by a 256-bit ECC protocol. A 256-bit ECC key size ensures a high level of security comparable to a 2072-bit RSA key size – the former encryption standard [24], [60]. Furthermore, the `secp256k1` elliptic curve proposed is already used by most blockchains [42]. Hence, there is no need for external third parties to define the system cryptographic parameters [25], which preserves the security of the FISIE system.

2) Lookup Tables: For a particular IoT device or fog node, its address on the lookup table is its blockchain address, which is generated from the hash of its public key. When device A communicates with device B, B verifies the address of A on the lookup tables by extracting the public key of A from its signature, which is generated by its secret key. That is, some third entity C cannot impersonate A unless C possess the secret key of A. Hence, the security of the mutual authentication and identity verification, is equivalent to the security of the 256-bit ECC protocol used [24].

3) Payment: All payments are recorded on the blockchain via IIMSC. Hence, any disputes that may arise can be verified against the blockchain to ensure the accuracy of all claims. Furthermore, since all entities are registered with IIMSC, future implementations could freeze IoT assets until a dispute is resolved, or add a reputation score to IoT devices to track how often they default on service payments.

4) Ring Signature: An oracle uses an IoT address a_ω to pose as an IoT device when interacting with a fog node. The oracle then submits its audit results to IIMSC using an oracle address a_Ω. The oracle audit submission includes a ring signature, which is a zero-knowledge proof of membership to prove the audit was done with an IoT address, without revealing which. Since all blockchain records are public, fog nodes can see which IoT devices are in the ring, and therefore could potentially be the oracle. The probability that any IoT address in the ring belongs to the oracle is $1/n$ for a ring of

n IoT addresses, which diminishes as n increases. However, a larger n requires more logging space on the blockchain and more verification computation from IIMSC, which could be costly. Hence, a balance is required to increase n as much as is reasonable. Alternatively, oracle o may send a ring signature $\mathcal{R}_\omega$ with a large n to a secondary computing oracle such as TrueBit[6] for transparent verification. The computing oracle then sends the result to IIMSC. This process would allow for larger ring signatures, thus increasing audit security, without inflating the cost of the smart contract.

B. *Auditing sampling policies*

For a set of fog nodes F and a size $\mathcal{C}$, we sample a distinct subset $F_\mathcal{C}$ from F of $\mathcal{C}$ nodes. Service audits are executed over members in $F_\mathcal{C}$ either sequentially or simultaneously. The purpose of clustering fog nodes even when executing sequentially is to limit the number of repeated audits to the same fog node in a short time period. That is, a larger size $\mathcal{C}$ increases the expected time between audits to the same fog node. Each sampling policy defines a specific way that clusters are sampled.

1) Random sampling: Clusters of size $\mathcal{C}$ are sampled randomly without repetition from the set F. This sampling method weighs all members equally, and makes no distinction between fog nodes who have previously passed their service audits, and those who haven't.

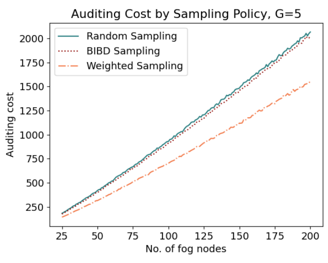
2) Weighted sampling: This sampling assigns a weight to each fog node, based on their previous audit history. The weights are initialized to be uniform. If a fog node fails a service audit, the weight is increased, denoting that this fog node should be audited more often. Similarly, if a fog node passes a service audit, we decrease the weight, indirectly giving audit priority to all other fog nodes. In practice, this method would require an oracle to keep a separate internal log of fog audit history.

3) BIBD sampling: This method is based on combinatorial design theory [61] where we build a series of finite balanced sets [62] of fog nodes F_B. We define the (F,B, B)-balanced incomplete block design (BIBD) as a series of blocks, i.e., subsets of F, that are of size B and have each fog node appear in B distinct blocks. The (F,B, B)-BIBD is computed at the beginning of the simulation, and is re-computed every time a fog node is ejected from the system.

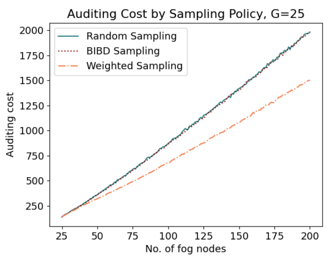
C. *Auditing cost*

We assign each fog node $f \in F$ a random 'malicious rate' $m_f \in [0.4, 1]$, a probability that the next request response will be faulty. By the definition of `Fog_penalize`, a fog node penalty will deduct a portion of the fog deposit, and remove the fog node from the system once that deposit reaches 0. For our simulation, we set each fog deposit to 3 and enforce a penalty of -1. That is, a fog node is removed from the system if it fails 3 service audits. For this simulation, we suppose the malicious rate of each fog node does not change in response to an audit result. The simulation is executed 1000 times per

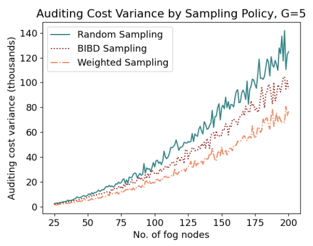
[6]https://truebit.io/



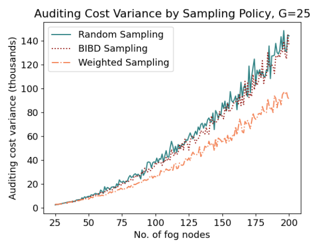
(a) Mean, $G = 5$



(b) Mean, $G = 25$



(c) Variance, $G = 5$



(d) Variance, $G = 25$

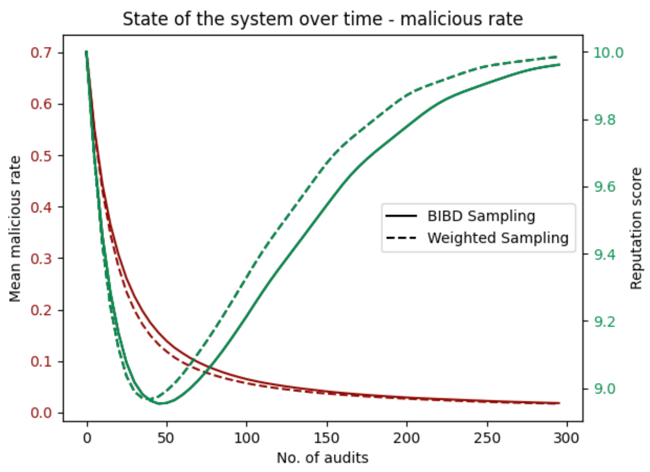
Figure 5: The means and variances of auditing costs necessary to expel all malicious nodes.

audit scheduling policy. We show the average of the results in Fig. 5a and 5b, and the variance of the results in Fig. 5c and 5d. From the results, the weighted sampling method consistently outperforms both random sampling and BIBD sampling in both mean and variance. Between random and BIBD methods, BIBD sampling performs marginally better. It is noted that using a larger cluster size $\mathcal{C}$ has no significant effect on the auditing cost.

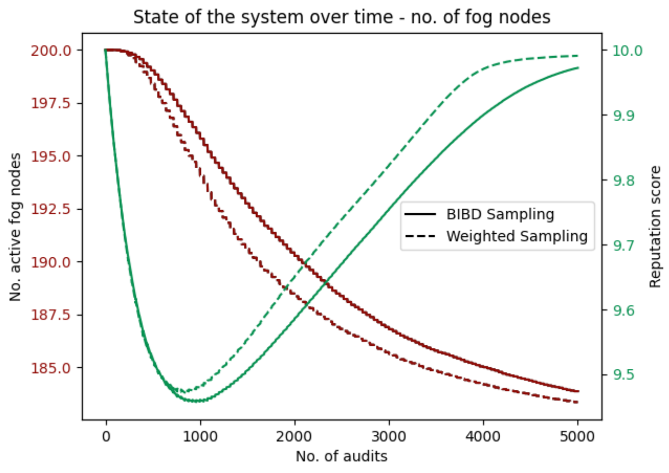
D. *State of the system*

Now we suppose that a fog node may alter it's malicious rate in response to a service audit result. If a fog node fails a service audit, we decrease the malicious rate by a random fraction. Two scenarios may result: 1) the fog node adjusts its malicious rate slowly towards 0 and redeems its reputation score, thus staying in the system, or 2) the fog node fails more service audits before fully redeeming its reputation score, and is ejected from the system. In both cases, the integrity of the overall system increases. We observe this trade-off between the number of malicious nodes and the integrity of the system by simulating the state of the system over time. For these simulations, we set IIMSC.$R_{\texttt{Min}} = 0$, and IIMSC.$R_{\texttt{Init}} =$ IIMSC.$R_{\texttt{Max}} = 10$.

1) By malicious rate: When a fog node fails a service audit, we decrease its malicious rate. Therefore, we expect the overall malicious rate to decrease over time. If a fog node is ejected from the system, then only fog nodes with lower malicious rates would stay in the system, further supporting our hypothesis. Indeed, as seen in Fig. 6a, there is a significant drop in the malicious rate over the first several audits. As expected, the overall reputation score initially drops, but slowly recovers once the majority of the fog nodes begin to behave properly.



(a) By mean malicious rate



(b) By number of fog nodes

Figure 6: The integrity of the FISIE system over time

The redemption of the overall reputation score of the system is quicker with the weighted sampling method.

2) By number of fog nodes: Though not as drastic of a drop as the malicious rate, we do observe a decrease in the number of total fog nodes. Over time, fog nodes who have not sufficiently decreased their malicious rate are ejected from the system. A smaller pool of fog nodes, mostly composed of honorable fog nodes, will increase the total average reputation score. This is seen in the direct trade-off between increased reputation score and decreased number of fog nodes in Fig. 6b. The decrease in the number of active fog nodes is more drastic with the weighted sampling method. Over both simulation results, it is clear that the Weighted sampling method reaches full integrity in a shorter amount of time.

XI. Future Work and Conclusions

A key aspect of IoT security is ensuring the integrity of the fog nodes that interact with IoT. In this paper, we proposed a general architecture for heterogeneous IoT and blockchain-enabled fog nodes. We defined a smart contract-based system for mutual authentication, monetization, and fog integrity enforcement. Finally, we analyzed the security of our system, and analysed the simulation results of our proposed service auditing over several audit scheduling policies. We found the weighted sampling method to increase system integrity in fewer total audits, hence lower blockchain cost.

In this study, the construction and analysis of the proposed system is theoretical. In future work, we will build a Proof of Concept model to study the feasibility of blockchain-enabled fog nodes, the actual incurred latency of mutual authentication and IoT task processing, and the actual behaviour of fog nodes over time. Furthermore, we will test various audit scheduling policies based on the real-time results of the system. Finally, we will include a data auditing mechanism in a public system to expand the scope of fog integrity. In a public IoT-fog environment with decentralized fog devices, integrity enforcement of fog will keep the environment safe and stable for IoT, and enable the expansion of IoT applications with the full cooperation of fog towards a real-world smart city [2].

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [CGS D - 558695 - 2021].

References

[1] S. Kumar, P. Tiwari, and M. Zymbler, "Internet of things is a revolutionary approach for future technology enhancement: a review," *Journal of Big data*, vol. 6, no. 1, pp. 1–21, 2019.
[2] C. Zhang, "Design and application of fog computing and internet of things service platform for smart city," *Future Generation Computer Systems*, vol. 112, pp. 630–640, 2020.
[3] CISCO, "Fog computing and the internet of things: Extend the cloud to where the things are," tech. rep., 2015. Accessed: 2021-09.
[4] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn, *et al.*, "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures," *Journal of Cloud Computing*, vol. 7, no. 1, p. 16, 2018.
[5] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149–160, 2019.
[6] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, "A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing," in *2015 International Conference on Information Networking (ICOIN)*, pp. 324–329, IEEE, 2015.
[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.
[8] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE access*, vol. 5, pp. 9882–9910, 2017.
[9] A. Giordano, G. Spezzano, and A. Vinci, "Smart agents and fog computing for smart city applications," in *International Conference on Smart Cities*, pp. 137–146, Springer, 2016.
[10] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management and evaluation of fog computing systems: A survey," *IEEE Internet of Things Journal*, 2020.
[11] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–38, 2022.
[12] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2020.

[13] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future generation computer systems*, vol. 82, pp. 395–411, 2018.

[14] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, "A blockchain-based mutual authentication scheme for collaborative edge computing," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 146–158, 2021.

[15] L. Qiao, Y. Li, F. Wang, and B. Yang, "Lightweight integrity auditing of edge data for distributed edge computing scenarios," *Ad Hoc Networks*, vol. 133, p. 102906, 2022.

[16] J. Wang, L. Wu, K.-K. R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1984–1992, 2019.

[17] M. Debe, K. Salah, M. H. U. Rehman, and D. Svetinovic, "Monetization of services provided by public fog nodes using blockchain and smart contracts," *IEEE Access*, vol. 8, pp. 20118–20128, 2020.

[18] H. Huang, X. Chen, Q. Wu, X. Huang, and J. Shen, "Bitcoin-based fair payments for outsourcing computations of fog devices," *Future Generation Computer Systems*, vol. 78, pp. 850–858, 2018.

[19] G. Liu, J. Wu, and T. Wang, "Blockchain-enabled fog resource access and granting," *Intelligent and Converged Networks*, vol. 2, no. 2, pp. 108–114, 2021.

[20] V. Buterin, "Ethereum white paper," tech. rep., 2013.

[21] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4682–4696, 2020.

[22] S. Algarni, F. Eassa, K. Almarhabi, A. Almalaise, E. Albassam, K. Alsubhi, and M. Yamin, "Blockchain-based secured access control in an iot system," *Applied Sciences*, vol. 11, no. 4, p. 1772, 2021.

[23] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of iot devices using blockchain-enabled fog nodes," in *2018 IEEE/ACS 15th international conference on computer systems and applications (AICCSA)*, pp. 1–8, IEEE, 2018.

[24] S. Kalra and S. K. Sood, "Secure authentication scheme for iot and cloud servers," *Pervasive and Mobile Computing*, vol. 24, pp. 210–223, 2015.

[25] S. Singh and V. K. Chaurasiya, "Mutual authentication scheme of iot devices in fog computing environment," *Cluster Computing*, vol. 24, pp. 1643–1657, 2021.

[26] A. A.-N. Patwary, A. Fu, S. K. Battula, R. K. Naha, S. Garg, and A. Mahanti, "Fogauthchain: A secure location-based authentication scheme in fog computing environments using blockchain," *Computer Communications*, vol. 162, pp. 212–224, 2020.

[27] Y. Ding, Y. Li, W. Yang, and K. Zhang, "Edge data integrity verification scheme supporting data dynamics and batch auditing," *Journal of Systems Architecture*, vol. 128, p. 102560, 2022.

[28] I. Zikratov, A. Kuzmin, V. Akimenko, V. Niculichev, and L. Yalansky, "Ensuring data integrity using blockchain technology," in *2017 20th Conference of Open Innovations Association (FRUCT)*, pp. 534–539, IEEE, 2017.

[29] H. Tian, F. Nan, C.-C. Chang, Y. Huang, J. Lu, and Y. Du, "Privacy-preserving public auditing for secure data storage in fog-to-cloud computing," *Journal of Network and Computer Applications*, vol. 127, pp. 59–69, 2019.

[30] A. Palai, M. Vora, and A. Shah, "Empowering light nodes in blockchains with block summarization," in *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*, pp. 1–5, IEEE, 2018.

[31] E. Reilly, M. Maloney, M. Siegel, and G. Falco, "An iot integrity-first communication protocol via an ethereum blockchain light client," in *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, pp. 53–56, IEEE, 2019.

[32] A. Hafid, A. S. Hafid, and M. Samih, "Scaling blockchains: A comprehensive survey," *IEEE access*, vol. 8, pp. 125244–125262, 2020.

[33] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, 2022.

[34] V. Kapoor, V. S. Abraham, and R. Singh, "Elliptic curve cryptography," *Ubiquity*, vol. 2008, no. May, pp. 1–8, 2008.

[35] E. Morais, C. van Wijk, and T. Koens, "Zero knowledge set membership," *none*, 2018.

[36] D. Vujičić, D. Jagodić, and S. Ranđić, "Blockchain technology, bitcoin, and ethereum: A brief overview," in *2018 17th international symposium infoteh-jahorina (infoteh)*, pp. 1–6, IEEE, 2018.

[37] O. Bouachir, M. Aloqaily, L. Tseng, and A. Boukerche, "Blockchain and fog computing for cyberphysical systems: The case of smart industry," *Computer*, vol. 53, no. 9, pp. 36–45, 2020.

[38] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa, and A. Bani-Hani, "Blockchain smart contracts: Applications, challenges, and future trends," *Peer-to-peer Networking and Applications*, vol. 14, pp. 2901–2925, 2021.

[39] A. Ouaddah, A. Abou El Kalam, and A. A. Ouahman, "Harnessing the power of blockchain technology to solve iot security & privacy issues.," in *ICC*, pp. 7–1, 2017.

[40] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: review, comparison, and open research challenges," *IEEE access*, vol. 8, pp. 85675–85685, 2020.

[41] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs*, vol. 1, pp. 1–136, 2021.

[42] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, pp. 36–63, 2001.

[43] M. Qu, "Recommended elliptic curve domain parameters," *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6*, 1999.

[44] H. Mayer, "Ecdsa security in bitcoin and ethereum: a research survey,"

[45] J. Thakur and N. Kumar, "Des, aes and blowfish: Symmetric key cryptography algorithms simulation based performance analysis," *International journal of emerging technology and advanced engineering*, vol. 1, no. 2, pp. 6–12, 2011.

[46] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pp. 105–110, IEEE, 2015.

[47] R. Mühlberger, S. Bachhofner, E. Castelló Ferrer, C. Di Ciccio, I. Weber, M. Wöhrer, and U. Zdun, "Foundational oracle patterns: Connecting blockchain to the off-chain world," in *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2020 Blockchain and RPA Forum, Seville, Spain, September 13–18, 2020, Proceedings 18*, pp. 35–51, Springer, 2020.

[48] H. Tran-Dang and D.-S. Kim, "A survey on matching theory for distributed computation offloading in iot-fog-cloud systems: Perspectives and open issues," *IEEE Access*, 2022.

[49] Y.-Y. Shih, C.-Y. Wang, and A.-C. Pang, "Fog computing service provision using bargaining solutions," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1765–1780, 2019.

[50] X. Peng, K. Ota, and M. Dong, "Multi-attribute based double auction towards resource allocation in vehicular fog computing," *IEEE Internet of Things Journal*, 2020.

[51] J. Metzger, "Decentralized justice in the era of blockchain," *IJODR*, vol. 5, p. 69, 2018.

[52] Z. Xu and L. Chen, "Div: resolving the dynamic issues of zero-knowledge set membership proof in the blockchain," in *Proceedings of the 2021 international conference on management of data*, pp. 2036–2048, 2021.

[53] Y. F. Chung, Z. Y. Wu, and T. S. Chen, "Ring signature scheme for ecc-based anonymous signcryption," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 669–674, 2009.

[54] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *2011 sixth annual ChinaGrid conference*, pp. 3–9, IEEE, 2011.

[55] J. M. Schopf and F. Berman, "Stochastic scheduling," in *Proceedings of the 1999 ACM/IEEE Conference on Supercomputing*, pp. 48–es, 1999.

[56] R. Kozhan and G. Viswanath-Natraj, "Decentralized stablecoins and collateral risk," *WBS Finance Group Research Paper*, 2021.

[57] H. Yu, Z. Shen, C. Miao, C. Leung, and D. Niyato, "A survey of trust and reputation management systems in wireless communications," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1755–1772, 2010.

[58] E. Bellini, Y. Iraqi, and E. Damiani, "Blockchain-based distributed trust and reputation management systems: A survey," *IEEE Access*, vol. 8, pp. 21127–21151, 2020.

[59] R. Govindaraj, P. Govindaraj, S. Chowdhury, D. Kim, D.-T. Tran, and A. N. Le, "A review on various applications of reputation based trust management.," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 10, 2021.

[60] A. Hamza and B. Kumar, "A review paper on des, aes, rsa encryption standards," in *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, pp. 333–338, IEEE, 2020.

[61] D. R. Stinson, *Combinatorial designs: constructions and analysis*, vol. 480. Springer, 2004.

[62] R. Bose and S. Shrikhande, "On the composition of balanced incomplete block designs," *Canadian Journal of Mathematics*, vol. 12, pp. 177–188, 1960.

Chapter 7

Conclusion

7.1 Summary of current work

Our research sought to formulate a scalable design & dimensioning scheme that constructs a general, robust and dynamic fog infrastructure. The first contributions formulates a scalable design & dimensioning scheme for a generalized static fog infrastructure. The fog infrastructure supports a high degree of multi-task IoT requests over a possibly congested network. The second contribution extends the static fog infrastructure to include on-demand fog nodes, which activate in light of a network overload. Such an overload may occur either from high network traffic, or from failed fog nodes that put stress on the remaining active nodes. The addition of on-demand fog nodes creates a robust fog infrastructure for a dynamic IoT-fog environment. The third contribution introduces fog-enabled UAVs to further support overloaded fog nodes and IoT devices in remote areas. The mobility of fog-UAVs is only restricted by nearby charging stations. We allow for fog-UAVs to move between charging stations, and therefore broaden their range of movement. The addition of mobile fog-UAVs creates a mobile and flexible fog infrastructure for a volatile IoT-fog environment. Finally, the fourth contribution enforces integrity onto fog nodes through a blockchain-based service auditing system. Penalties and incentives are applied to the fog node by the blockchain, based on the audit results. Hence, fog nodes are encouraged to act with integrity at all times, or risk being penalized or ejected from the system. Furthermore, the system enables secure communication between IoT and fog and streamlined IoT service payments to fog. This results in a fair and secure fog infrastructure for a trustless IoT-fog environment.

7.2 Impact

The fog-computing paradigm is the collaboration of distributed heterogeneous nodes on the network edge, capable of providing real-time communication and processing. The research considers (1) that the constructed fog infrastructure should be flexible and operable in any IoT environment, (2) that IoT traffic may be volatile and spread out over a large region, and (3) that the IoT-fog environment is trustless and fog nodes may be malicious. The resulting fog infrastructure keeps the strengths of fog-computing while addressing its weaknesses.

Much research has been done on the interactions between IoT, fog and Cloud on a pre-existing fog infrastructure [8]. However, a real-life implementation of an efficient fog infrastructure is dependent on the optimal design & dimensioning given the particular goals of the implementer. The focus of our research is to provide a blueprint of a general fog infrastructure over multiple considerations of fog reliability, mobility and integrity, as well as IoT traffic. Therefore, our generalized fog design & dimensioning scheme encourages the adoption, development and implementation of real-life fog infrastructures towards collaborative smart cities.

7.3 Limitations and Future Work

Several assumptions and simplifications over the IoT-fog network were made in order to concentrate the scope of the research. Further investigation is needed in order to verify to what degree our assumptions hold true, and extend our work to accommodate environmental restrictions. Hence, the current limitations of this thesis motivate a plan for future work.

7.3.1 IoT traffic distributions

In the presence of Human Dynamics, IoT traffic can be approximated to follow a Poisson distribution [36]. Thus, chapters 3 to 5 assume IoT traffic follows a Poisson distribution in our traffic estimations. For the estimation of network congestion, we once again assumed a Poisson arrival rate of IoT traffic to the network. However, not all IoT requests require fog assistance. Indeed, the true distribution and size of IoT requests that require fog support may be different than those used in our models.

In the case of IoT traffic, analysis over live IoT data is needed to verify actual IoT traffic patterns in both the best-case and worst-case. Similarly, analysis of actual network congestion and faulty transmission can be done to gauge the effects of network factors on the total round-trip latency of IoT requests.

7.3.2 Communication interference

In chapters 3 and 4, our models consider IoT request latency caused only by network congestion to and from the fog nodes. That is, we assumed all communication between IoT and the network edge were unimpeded by environmental interference or faults in IoT-fog communication. Indeed, it may be that IoT-fog communication suffers from similar building and environmental interference as Wi-Fi [43].

When working with fog-UAVs for IoT support such as in chapter 5, further considerations are needed in regards to outdoor IoT-fog communication. Since a fog-UAV is deployed and hovers over an IoT service area, we must consider interference caused by the fog-UAV hovering height. Furthermore, we assume that a fog-UAV can support all IoT devices within its service radius. However, only a subset of these IoT devices may have clear communication due to environmental factors.

7.3.3 Fog-UAV battery charge and mobility

For electric vehicles, the first ~80% of a battery charges approximately linear [44]. For simplicity, we extended this assumption to fog-UAVs, and assumed there was always at least one charged fog-UAV ready for deployment. However, the approximate charging function of a UAV may be different than an electric vehicle in practice. Furthermore, it may not be reasonable to assume there is always a charged fog-UAV ready for deployment. Indeed, a fog-UAV needs idle time to charge its battery, which was not considered in this first iteration of the fog-UAV set covering problem.

7.3.4 Mobility restrictions

In chapter 5, fog-UAV mobility is assumed to only be restricted by its battery. That is, we ignore any environmental obstacles and assume direct fog-UAV flight 'as the crow flies'. However, cities with tall buildings or mountainous regions may impede fog-UAV mobility, shortening its overall distance radius for the same battery charge.

The fog-UAV probabilistic location set covering problem assumes that all fog-UAVs deployed to service IoT have an ~80% charged battery, and therefore have the majority of their mobility potential. In reality, this may not be the case, and partial charging may be needed to ensure optimal IoT service. Future work for fog-UAVs includes a stochastic programming model that considers charging time, partial charging, and fog-UAV re-distribution among charging stations for future deployment. Discrete-event simulation can be used to verify the resulting fog-UAV infrastructure.

7.3.5 Blockchain-based integrity enforcement – cost and delay

In chapter 6, we defined a smart contract to streamline IoT-fog mutual authentication, monetization and integrity enforcement. The blockchain-based service auditing system is evaluated over a theoretical malicious rate and audit reaction of fog nodes. However, the cost of the smart contract functionalities, and in particular the service audits, were not explored in detailed. Since our implementation is generalized to any smart contract-capable blockchain, implementation costs may vary across blockchain platforms. Indeed, we recognize the value in evaluating the efficiency and cost of a live system. This will also allow us to evaluate the system over different audit scheduling policies to determine the optimal configuration for ensuring fog integrity.

Furthermore, the latency delay from mutual authentication was not explored in detail. Once again, the largest latency factor in mutual authentication is the communication delay which is affected by communication interference between IoT, fog, and the blockchain. Though blockchain-enabled fog nodes can minimize this latency, further exploration is needed with a Proof of Concept (PoC) implementation.

Isma's Note: what is the delay improvement when using blockchain-enabled fog nodes, and can we design a fog infrastructure with bc nodes

7.3.6 Theoretical formulation and simulation

The fog design & dimensioning scheme is formulated to support a high degree of IoT traffic. However, the optimal designed fog infrastructure solution is not simulated as a 'live' system to see to what degree the full fog infrastructure is used. Future work may include the implementation of discrete-even simulation over the resulting fog infrastructure. Such a simulation would verify if (1) the fog infrastructure is sufficient in supporting any volatile IoT environment, and (2) the resulting fog infrastructure is overambitious in size, having several sparingly used fog nodes. That is, a balance is needed between the cost of the fog infrastructure and the percentile of total IoT traffic that is supported.

7.3.7 End-to-end fog design & dimensioning

Each of our contributions related to fog design & dimensioning builds upon the previous contribution. On-demand fog node design, dimensioning and rerouting configurations were added onto an existing static fog infrastructure, and mobile fog-UAVs and UAV recharge stations were added onto an existing dynamic fog infrastructure. There may exist an optimal design & dimensioning configuration of static, on-demand and mobile fog nodes that has not been seen due to the layering of different models. Therefore, there exists an opportunity to combine the separate models into a master one-shot model. Furthermore, sensitivity analysis over the master model can be done to gauge the effects of future localized increases in IoT traffic on the resulting fog infrastructure.

7.3.8 Proof of Concept

The largest opportunity, and the purpose of our research, is to evaluate the effectiveness of a real-life fog infrastructure. A small scale Proof of Concept (PoC) of a fog infrastructure should be evaluated and analyzed to determine the effectiveness of our approach and whether there are any aspects of the IoT-fog environment that we have failed to consider. Indeed, the objective of our research is to facilitate the implementation of a live fog infrastructure. Hence, the construction and evaluation of a PoC would be the next step in realizing a cooperative IoT-fog-Cloud environment for real-life smart cities.

7.4 Retrospective

This section provides a retrospective of my time as a Master's and a PhD student. It details what I have learned and how my work has evolved throughout my studies.

7.4.1 Modeling

In chapter 3, I was given an initial MILP model, and Column Generation variant, which I modified slightly. In chapter 4, I extended the work from the previous chapter and continued to use Column Generation to scale the intractable MILP model. However, though faster, both Column Generation models created a heuristic solution. That is, the Column Generation solutions were not optimal. Though I was able to reduce its optimal cost by restricting the viable columns that could be generated per iteration, it became clear that Column Generation was not fit for the types of MILP models that were formulated. Indeed, suboptimal results are still results, since they indicate that a better method may exist.

In chapter 5, I used Benders Decomposition as the decomposition technique, which gave an equal optimal solution. It was also in this publication that I used pre-computation techniques to simplify viable constraints and simplify the model. Indeed, this was the first publication in which I formulated the model from scratch, and had more freedom to adjust the model during development. By this point, I had become comfortable with model formulations and relaxations.

CPLEX

All models were executed in C++ with IBM CPLEX Optimization Studio. I learned C++ at the beginning of my studies with the primary objective to write the model simulation codes in CPLEX. By coding my models, and observing the effects of certain types of constraints, integer versus continuous decision variables, and large variable sets on the execution time, I gained a solid understanding of how the theoretical MILP models translated to coded implementations. This motivated the pre-computed constraint simplifications in chapter 5, which was both easier to understand on paper, as well as quicker to execute in code. Certainly, since the eventual objectives of my formulated large-scale MILP models are to be implemented, great focus was given to making them easy to understand and hence straightforward to code.

‘Ideal’ configurations

Assumptions and simplifications were essential to containing the scope of the formulated models. Without them, the focus and objective of each model risks being lost on the reader. Though, at times these assumptions would create ‘ideal’ circumstances that may not be realistic, they provide a starting point for the current and future contributions.

In chapter 3, I attempted to model every component of IoT-fog interaction. In doing so, the scope of the problem became quite large. Indeed, it may have served the contribution better to separate the components of static fog design & dimensioning, and IoT traffic congestion estimation into two detailed contributions of their own.

In chapter 5, I did not consider environmental interference or communication loss from a high-altitude fog-UAV. In other words, ‘ideal’ circumstances were assumed for IoT-UAV communication. However, this allowed the focus of the manuscript to be on the probabilistic location set covering of fog-UAVs. Furthermore, this invites for future contributions that focus purely on the reliability of IoT-UAV communication in different environments.

7.4.2 Simulation

In each of chapters 3, 4 and 5, simulation was used to validate the efficiency of the different models in terms of execution time, optimal solution cost and optimal fog design. In chapter 3, larger candidate networks were built by simply scaling the resources of previous iterations. Starting in chapter 4, I began to add more noise and variability into the initial configurations used for each iteration of the simulation. This increased the breadth of configurations that the models were compared against, giving a more accurate view of the efficiency of the relaxed, decomposed or heuristic models against the exact MILP model.

Up until now, discrete-event simulation has not been used to validate the optimal fog infrastructures. Such a simulation would help gauge the practical applicability of the proposed methods. In particular, when dynamic and/or mobile fog nodes are used, discrete-event simulation can be useful to analyze when dynamic fog nodes are needed, or the charging statuses of mobile fog nodes when they are deployed. Furthermore, current models have considered a ‘worst-case’ IoT traffic scenario when modeling the optimal fog infrastructure. Hence, there exists a future opportunity to continue this work in fog design & dimensioning using different simulation techniques and evaluation metrics to further validate or update current models.

7.4.3 Blockchain

At the beginning of my Master’s, I knew very little regarding blockchain, cryptography, and zero-knowledge proofs – topics that became the core of chapter 6. Indeed, enforcing the security and integrity of the IoT-fog environment is equally as important to my other contributions when considering their applicability to smart cities. As my last major contribution of this thesis, I had become comfortable with how to research and learn new topics, and the learning curve on this last contribution was significantly lower than that of my first contribution.

7.4.4 Research Skills Development

Overall, since my first contribution in the summer of 2019, I am far more comfortable and confident with finding supporting research, with identifying opportunities among the state-of-the-art, and with writing scientific papers. Initially, I relied heavily on the guidance of my supervisors to help define my research. My first two contributions in particular in chapters 2 and 3 went through multiple rewrites, updates and revisions with countless hours of support from my supervisors. By the last two contributions in chapters 5 and 6, I had taken a more self-directed approach, with minimal direction from my supervisors as I had taken more ownership of my research. With each contribution, my supervisors entrusted me with greater autonomy, as I had shown to have a clear vision, and a better grasp of the material. I am grateful for the increasing trust and autonomy my supervisors gave me throughout my PhD journey, which assisted in my growth as a researcher.

7.4.5 Teaching & Time Management

Throughout the last couple of years, I had the privilege to work as a Teaching Assistant, and later as a Lecturer. These roles helped me to become comfortable and confident in front of a classroom, and to become organized and attentive to detail in lesson preparation. These teaching opportunities translated to my research, as I learned to write concisely and clearly.

However, these teaching roles added to my overall workload, forcing me to learn to manage my time well between teaching and research. Though this past semester has been hectic and overwhelming at times, it has also been profoundly rewarding. It has served as an excellent exercise in prioritization, which will no doubt be helpful in future professional settings.

7.4.6 Problem Solving & Perseverance

None of the contributions of this thesis were straightforward. Embracing more autonomy of my research also left it to me to find my research direction and come up with novel solutions to my problems. Though some of these took me a few days to figure out, others would take me several months of refinement to properly define. For example, chapter 5 went through several iterations before I was satisfied with the problem statement and the model formulation. However, this was excellent in developing my problem solving skills, and taught me perseverance. Indeed, when I found myself at an impasse in my research, I learned to take some time away from it, then come back to the problem from a different angle. This technique was instrumental in helping me overcome hurdles and evenutally complete my thesis.

7.4.7 Future Outlook

I view the collection of publications in this thesis as merely a starting point for optimal fog design & dimensioning for IoT support in smart cities. As discussed in chapter 7.3, many opportunities exist as future work to continue this line of research.

It is clear to me that, though there still remains much work in the theoretical modelling of the fog infrastructure, the natural next step towards real-life implementations is a small-scale Proof of Concept (PoC). Such a PoC would gather new insights into the optimal design & dimensioning of a live fog infrastructure. Furthermore, continued research with a PoC could potentially lead to live large-scale implementations in smart cities.

Bibliography

[1] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 32, 2017.

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

[3] A. Giordano, G. Spezzano, and A. Vinci, "Smart agents and fog computing for smart city applications," in *International Conference on Smart Cities.* Springer, 2016, pp. 137–146.

[4] S. Al-Sarawi, M. Anbar, R. Abdullah, and A. B. Al Hawari, "Internet of things market analysis forecasts, 2020–2030," in *2020 Fourth World Conference on smart trends in systems, security and sustainability (WorldS4).* IEEE, 2020, pp. 449–453.

[5] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops).* IEEE, 2015, pp. 105–110.

[6] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing.* ACM, 2012, pp. 13–16.

[8] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management, and evaluation of fog computing systems: a survey," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2494–2516, 2020.

[9] C. Chang, S. N. Srirama, and R. Buyya, "Indie fog: An efficient fog-computing infrastructure for the internet of things," *Computer*, vol. 50, no. 9, pp. 92–98, 2017.

[10] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData & SocialInformatics 2015.* ACM, 2015, p. 28.

[11] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "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*, vol. 47, no. 9, pp. 1275–1296, 2017.

[12] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems.* ACM, 2016, pp. 258–269.

[13] N. Chen, Y. Yang, T. Zhang, M.-T. Zhou, X. Luo, and J. K. Zao, "Fog as a service technology," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 95–101, 2018.

[14] G. L. Santos, P. T. Endo, M. F. F. da Silva Lisboa, L. G. F. da Silva, D. Sadok, J. Kelner, T. Lynn *et al.*, "Analyzing the availability and performance of an e-health system integrated with edge, fog and cloud infrastructures," *Journal of Cloud Computing*, vol. 7, no. 1, p. 16, 2018.

[15] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2015.

[16] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[17] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor–critic deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061–2073, 2018.

[18] M. Taneja and A. Davy, "Resource aware placement of iot application modules in fog-cloud computing paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM).* IEEE, 2017, pp. 1222–1228.

[19] E. Yigitoglu, M. Mohamed, L. Liu, and H. Ludwig, "Foggy: a framework for continuous automated iot application deployment in fog computing," in *2017 IEEE International Conference on AI & Mobile Services (AIMS).* IEEE, 2017, pp. 38–45.

[20] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource provisioning for iot services in the fog," in *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA).* IEEE, 2016, pp. 32–39.

[21] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "Mist: Fog-based data analytics scheme with cost-efficient resource provisioning for iot crowdsensing applications," *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[22] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: A simulator for virtual machine migration in fog computing," in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, 2017, pp. 47–52.

[23] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*, 2013, pp. 15–20.

[24] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future generation computer systems*, vol. 82, pp. 395–411, 2018.

[25] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4682–4696, 2020.

[26] S. Singh and V. K. Chaurasiya, "Mutual authentication scheme of iot devices in fog computing environment," *Cluster Computing*, vol. 24, pp. 1643–1657, 2021.

[27] S. Kalra and S. K. Sood, "Secure authentication scheme for iot and cloud servers," *Pervasive and Mobile Computing*, vol. 24, pp. 210–223, 2015.

[28] O. Bouachir, M. Aloqaily, L. Tseng, and A. Boukerche, "Blockchain and fog computing for cyberphysical systems: The case of smart industry," *Computer*, vol. 53, no. 9, pp. 36–45, 2020.

[29] D. Vujičić, D. Jagodić, and S. Ranić, "Blockchain technology, bitcoin, and ethereum: A brief overview," in *2018 17th international symposium infoteh-jahorina (infoteh)*. IEEE, 2018, pp. 1–6.

[30] G. Cheng, Y. Chen, S. Deng, H. Gao, and J. Yin, "A blockchain-based mutual authentication scheme for collaborative edge computing," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 146–158, 2021.

[31] J. Wang, L. Wu, K.-K. R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1984–1992, 2019.

[32] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of iot devices using blockchain-enabled fog nodes," in *2018 IEEE/ACS 15th international conference on computer systems and applications (AICCSA)*. IEEE, 2018, pp. 1–8.

[33] B. Li, Q. He, F. Chen, H. Jin, Y. Xiang, and Y. Yang, "Auditing cache data integrity in the edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1210–1223, 2020.

[34] L. Qiao, Y. Li, F. Wang, and B. Yang, "Lightweight integrity auditing of edge data for distributed edge computing scenarios," *Ad Hoc Networks*, vol. 133, p. 102906, 2022.

[35] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 149–160, 2019.

[36] A. J. Jara, D. Genoud, and Y. Bocchi, "Big data in smart cities: from poisson to human dynamics," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2014, pp. 785–790.

[37] C. ReVelle and K. Hogan, “A reliability-constrained siting model with local estimates of busy fractions,” *Environment and Planning B: Planning and Design*, vol. 15, no. 2, pp. 143–152, 1988.

[38] J.-F. Cordeau, F. Furini, and I. Ljubić, “Benders decomposition for very large scale partial set covering and maximal covering location problems,” *European Journal of Operational Research*, vol. 275, no. 3, pp. 882–896, 2019.

[39] I. Martinez, A. Jarray, and A. S. Hafid, “Scalable design and dimensioning of fog-computing infrastructure to support latency-sensitive iot applications,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5504–5520, 2020.

[40] I. Martinez, A. S. Hafid, and M. Gendreau, “Robust and fault-tolerant fog design & dimensioning for reliable operation,” *IEEE Internet of Things Journal*, 2022.

[41] ——, “Design and dimensioning of a uav set covering in high-traffic iot-fog environments,” *IEEE Internet of Things Journal*, 2024, submitted.

[42] ——, “A blockchain-based audit mechanism for trust and integrity in iot-fog environments,” *IEEE Transactions on Industrial Informatics*, 2024, submitted.

[43] A. Kashyap, U. Paul, and S. R. Das, “Deconstructing interference relations in wifi networks,” in *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. IEEE, 2010, pp. 1–9.

[44] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, “The electric vehicle routing problem with nonlinear charging function,” *Transportation Research Part B: Methodological*, vol. 103, pp. 87–110, 2017.