Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5331712, 15 pages
https://doi.org/10.1155/2022/5331712

Research Article

# OffFog: An Approach to Support the Definition of Offloading Policies on Fog Computing

**Sávio Melo**,[1] **Felipe Oliveira**,[1] **Cícero Silva**,[2] **Paulo Lopes**,[1] **and Gibeon Aquino**[1]

[1]*Department of Informatics and Applied Mathematics (DIMAp), Federal University of Rio Grande do Norte, Natal, 1524 RN, Brazil*
[2]*Federal Institute of Education, Science and Technology of Paraíba, Catolé do Rocha, 227 PB, Brazil*

Correspondence should be addressed to Sávio Melo; saviorennan@ufrn.edu.br

Received 15 July 2021; Revised 25 November 2021; Accepted 8 December 2021; Published 4 January 2022

Academic Editor: Pengfei Wang

IoT devices deployed in Smart Cities usually have significant resource limitations. For this reason, offload tasks or data to other layers such as fog or cloud is regularly adopted to smooth out this issue. Although data offloading is a well-known aspect of fog computing, the specification of offloading policies is still an open issue due to the lack of clear guidelines. Therefore, we propose OffFog—an approach to guide the definition of data offloading policies in the context of fog computing. In order to evaluate OffFog, we extended the well-known simulator *iFogSim* and conducted an experimental study based on an urban surveillance system. The results demonstrated the benefits of implementing data offloading based on OffFog recommended policies. Furthermore, we identified the best configuration involving design decisions such as data compression, data criticality, and storage thresholds. The best configuration produced at least 76% improvement in network latency and 5% in the average execution time compared to the *iFogSim* default strategy. We believe these results represent a significant step towards establishing a systematic decision framework for data offloading policies in the context of fog computing.

## 1. Introduction

Fog computing plays an essential role in building a sustainable IoT infrastructure for smart cities [1]. Many research efforts have been made on fog computing in smart cities, but there are still several open challenges towards its concrete realization [1, 2]. Moreover, despite Fog computing brings advantages to the development of IoT applications such as latency improvement, new challenges related to storage, and local processing power emerges in the context of Smart Cities [3, 4]. Indeed, IoT devices deployed in Smart Cities usually have significant resource limitations. Therefore, some strategies must be used to mitigate the problems caused by this issue. A usual solution is to transfer the responsibility to entities with more resources, known as offloading. One particular category is the data offloading between the fog and cloud. The need to transfer the data to the cloud emerges in several cases, being very usual when there is the need for long-term storage of data produced on the network border [5].

Data offloading affords several benefits to IoT applications, including energy-saving, storage reduction, decreased network usage, reduced decision-making time, among others. However, the correct execution of this task requires following specific rules (or policies) related to transferring data between persistence entities. These policies should be defined based on criteria such as load balancing, data volume, long-term storage, latency, data management and privacy, among others [5, 6]. Furthermore, they are context-dependent, i.e., they are strongly influenced by the application domain and the characteristics of the fog devices. Thus, defining and implementing effective data offloading strategies in the fog context is a challenging issue.

Therefore, our work proposes an approach to assist the definition of data offloading strategies in fog contexts. This approach is called OffFog and aims to facilitate decisions for selecting data offloading criteria. It consolidates the main concerns related to data offloading in fog contexts and offers proper guidelines in the form of a set of activities aiming at conducting the solution's architect to an effective offloading strategy. To the best of our knowledge, no other previous work provides clear guidelines on how to perform the data offloading process in fog computing-based applications.

As a case study, we use OffFog in the context of an urban security camera application, basing on the scenario proposed by Gupta et al. [7]. This case study attempts to demonstrate the feasibility of using OffFog in applications based on fog computing and, at the same time, to verify the offloading criteria of selected data influence in this scenario. For this, we extended the work of Naas et al. [8], including new offloading capabilities, such as data compression, and performed simulations in order to evaluate the impact of different levels of choice for these criteria.

The main contribution of this paper is threefold. (1) A taxonomy that compiles and structures the knowledge regarding concepts and techniques of data offloading in fog contexts. (2) The OffFog approach aiming to guide the definition of data offloading policies in the context of fog computing. (3) The extension of the well-known simulator *iFogSim* [7], including new data offloading capabilities. We believe these results represent a significant step towards establishing a systematic decision framework for data offloading policies in the context of fog computing.

The remainder of this article is organized as follows: Section 2 describes the related works. Section 3 presents the background related to data offloading on Fog computing. Section 4 describes the OffFog approach. A case study is described in Section 5, aiming at demonstrating the execution of OffFog in a scenario of a Surveillance Camera System (SCS). Section 6 presents the results of an experimental study involving simulations of the SCS scenario. Finally, Section 7 discusses the concluding remarks.

## 2. Related Works

Studies related to this work can be divided into two groups: studies dealing with data offloading in fog computing (Section 2.1); solutions that enable the simulation of applications based on fog computing (Section 2.2).

*2.1. Data Offloading.* According to Farahani et al. [9], Silva [10], Silva and Aquino Jr. [11], there are several challenges related to managing stored data in the fog layer. Furthermore, the storage capacity on Fog is usually limited when compared to Cloud [3, 11]. Thus, several works propose strategies to deal with these issues.

Silva et al. [12] propose a fog computing-based architecture to manage patients' medical data. In this architecture, the authors define a component particularly designed to store metadata related to medical records kept in the fog. Thus, this metadata helps decide which data to free or keep on the fog nodes. However, the proposed solution defines a generic behaviour without specify the offloading process's concrete rules (or policies).

Alelaiwi [13] and Pisani et al. [14] use the data offloading process to meet the requirements of their applications better. Thus, these authors consider strategies and policies that set the course for such a process, serving to mitigate the limitations of the fog environment. On the other hand, considering a dynamic fog computing network, Shan et al. [15] use the heterogeneity of sensor nodes as contextual information to promote the data offloading between sensors and fog nodes while improving the algorithm for stable performance in urban areas environments.

The data offloading process in fog computing is a topic of constant discussion. Alelaiwi [13] uses machine learning algorithms that select the data to be sent to the cloud. In this approach, the offloading of edge data is only performed after carrying out deep learning techniques. Pisani et al. [14] consider data that needs to be balanced across fog nodes to data offloading according to available storage. However, these studies do not discuss criteria related to the volume and nature of the data.

Aazam et al. [5] classify the types of offloading that exist, defining different situations where the process can occur and showing its variations. He et al. [16] consider that security and load balancing aspects are significant in the data offloading process. However, neither of the two works relates the properties of data and infrastructure.

Still dealing with security, He et al. [16] propose a security model for storing fog data, highlighting the possible vulnerabilities that this data may be subject to. Verma et al. [17] propose a processing balancing model adaptable to the context of fog data. However, the authors do not specify the techniques used in the processes and their possible impacts.

Mukherjee et al. [18] present a fog data offloading strategy to minimize energy consumption and the total delay related to the processing task for the end-user. The work formulates an optimization problem and provides a solution through the CVX tools, a modelling system based on MATLAB for convex optimization [18]. The simulation results demonstrated that the proposed solution offers minimal end-to-end latency than running all tasks on end-user devices and running all tasks on primary fog nodes. However, the study does not discuss criteria related to data volume and nature.

Zhu et al. [19] propose task offloading policy to help decide whether to process tasks locally and send them to the appropriate fog node or cloud. This approach considers calculating the execution time and energy consumption when running the task on the device compared to the execution time and energy consumed when transferring and receiving the processed task at the appropriate fog node. Dinh et al. [20] propose an offloading framework to optimize the task allocation decision and the allocation of computational resources.

In summary, several works in the literature propose solutions for data offloading in fog, which addresses different problems. The work of Bali et al. [21], for instance, provides a systematic literature review (SLR) where approaches on data offloading in IoT networks at edge and fog nodes can be found. In that same work, the authors propose an intelligent architecture for data offloading and propose a data offloading workflow for industrial IoT. However, there are no clear directions regarding the selection criteria to assist data offloading decision making. Therefore, unlike other works, this paper classifies the criteria of the data offloading process to guide in defining policies to enable the execution of data offloading in fog-based solutions.

*2.2. Simulation of Fog Computing Environments.* Some tools simulate fog computing environments. Margariti et al.'s [22] work investigates the simulation tools used to develop fog-based solutions. Among the simulation tools investigated, *iFogSim* stands out. *iFogSim* (https://github.com/cloudslab/ifogsim) is currently the most used tool to simulate fog computing-based environments. Described in Gupta et al. [7], this simulator implements resource management techniques on fog computing environments in order to measure latency and throughput.

Some *iFogSim* extensions were implemented, aiming to add additional features to this simulator. Lopes et al. [23] developed *MyiFogSim* (http://www.lrc.ic.unicamp.br/fogcomputing), an extension of *iFogSim* to support the mobility requirement of fog computing-based applications. In addition, Naas et al. [24] extended *iFogSim* to simulate scenarios with strategies aimed at optimizing data placement, called *iFogSimWithDataPlacement* (https://github.com/medislam/iFogSimWithDataPlacement).

An important extension of *iFogSim* that allows simulating environments with the requirement of mobility is the so-called *MobFogSim*. Proposed by Puliafito et al. [25], this extension allows overcoming the limitations of environments that require mobility. It also extends *iFogSim* to allow device mobility modelling. However, its distinguishing feature is support for service migration in fog computing.

Another well-known simulation environment for fog computing is the Edge-Fog cloud simulator [26]. This simulator is a Python implementation consisting of two layers: an outer layer consisting of the edge devices and an inner layer consisting of the fog devices. Furthermore, this implementation focuses on implementing the Least Processing Cost First (LPCF) algorithm, which reduces processing time and network costs.

Overall, none of the implementations mentioned above brings the possibility of simulating data offloading policies. Thus, *iFogSimWithDataPlacement* was extended in our work, generating a new implementation called *iFogSimOffload* (https://github.com/labcomu/iFogSimOffload), which differs from the others by providing some techniques related to the data offloading processes, such as compression and data criticality.

## 3. Data Offloading in Fog

The offloading process is responsible for performing migration of data or processing tasks, helping devices with limited computational capabilities [27]. Regarding data offloading, this is a technique that has been widely discussed in the fog computing paradigm [5]. Furthermore, it is generally applied due to the resource constraints of the entities that make up the fog. Data offloading is controlled through rules, which indicate how data is managed during this process. Finally, these rules are commonly called policies.

To organize the essential concepts involved in the data offloading process, we defined taxonomy. In particular, it compiles essential aspects for data offloading policies in the fog contexts. Moreover, the taxonomy describes the main properties to be considered when designing data offloading policies for the most diverse fog scenarios, considering the characteristic of the data and the infrastructure of the environments involved. Therefore, the section classifies some concepts about the data offloading process and exemplifies the main techniques used in such a process. Thus, this classification can serve as a basis for selecting techniques to define data offloading policies since it provides many of the concepts necessary for this definition.

*3.1. Taxonomy.* In our taxonomy, the criteria were divided into two main views: data and environment. Therefore, the classes and subclasses that compose the taxonomy are distributed in these views.

The Data View brings together the characteristics directly related to data generation, transmission, and processing in a fog computing environment. Thus, it is formed by the Security, Management, Nature, and Volume classes. On the other hand, the Environment View encompasses the physical characteristics related to the data offloading process in a fog environment. This view is related to the requirements of the environment in which the data resides, that is, the physical properties of the fog nodes. Therefore, it is formed only by the Infrastructure class.

Figure 1 shows the taxonomy proposed by Melo et al. [6]. It summarizes the classes that help design policies for data offloading in fog computing. Therefore, these classes will be detailed below.

*3.1.1. Security.* The use of inappropriate approaches creates many security problems in the nodes that form the fog [28]. However, applying the four pillars of information security (confidentiality, availability, integrity, and privacy) guarantees the security of information transferred in the fog.

As far as confidentiality is concerned, the application of authentication protocols guarantees the identity of the fog connected nodes [29]. In addition, the use of this type of protocol can prevent unauthorized people from accessing the data transmitted between these nodes. On the other hand, technologies that promote data availability must be applied to ensure that fog nodes or cloud servers are always available [29]. The use of techniques such as data replication and load balancing can help achieve this goal.

In the same context, data integrity ensures the detection of any unauthorized attempts to modify them. In fog computing, Blockchain technology is often used to avoid this type of problem [30]. The privacy attribute is intended to protect sensitive personal information [31]. To meet this requirement, [32] propose a Blockchain-based method to ensure the patient's medical data privacy in the fog computing environment. In this perspective, it is common to use Blockchain technology to provide the data anonymization feature.

*3.1.2. Management.* Transferring data from one device to another is a common task in fog computing. Therefore, this process needs to be well managed [5]. The primary motivation to consider this aspect in the data offloading process is that fog nodes are geographically distributed, making it challenging to identify the location of the data. In addition, several techniques can be used during data management, such as optimization and replication. Thus, using these

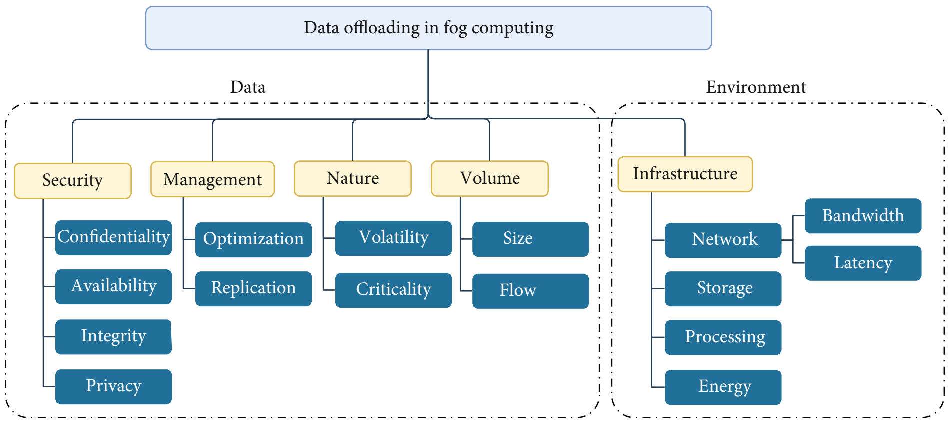


Figure 1: Taxonomy for data offloading in fog computing.

techniques will help distribute the data between the fog nodes more efficiently since the requested data may not be available in nearby nodes.

About data optimization, this process is directly related to the storage capacity and processing constraints of a fog node [33]. According to Rahmani et al. [34], two optimization techniques stand out in the fog computing paradigm: data filtering and compression.

*3.1.3. Nature.* Generally, environments in which fog computing is applied do not use their hardware or communication technologies as a solution for nonvolatile data storage [35]. On the other hand, most resources (storage and network) are provided by devices near the edge of the network, such as smartphones, smart TVs, or vehicles. Thus, this paradigm faces a challenge linked to the resources provided by its nodes, as they may at some point become unavailable for access [36].

In addition, many applications make use of fog computing to meet their real-time operational needs. At the same time, large-scale computing infrastructures are increasingly dealing with critical data, which is also stored in cloud databases [37]. Therefore, when operating/transmitting the data generated by the fog layer device, it is necessary to consider its different degrees of criticality. For example, critical data may be found in the scenario of monitoring patient health information. In contrast, some other data in the same solution may be less relevant and volatile.

*3.1.4. Volume.* The volume class is related to two aspects: data size and flow [38]. These two aspects are the most mentioned in the literature and the most relevant when the data analysis is done in the nodes that make up the fog or in external entities. In this context, analyzing these two factors in this paradigm allows data to be transferred to the cloud only when necessary.

The size aspect is related to the amount of data that needs to be processed and stored in the fog. Thus, many researchers point to this feature as one of the essential requirements for meeting requests [39]. In addition, this aspect must be considered to mitigate the storage limitation characteristic of the nodes that make up the fog. Concerning data flow, this aspect is related to the intensity at which data is sent from a source to a fog node. Thus, the flow can occur in two ways: event-driven [40] or real-time [41]. In real-time, streaming data is streamed into the fog immediately after generation, resulting in nodes being overloaded. Therefore, understanding the size and flow of data in the fog during the data offloading process is essential to keep the system running smoothly.

*3.1.5. Infrastructure.* The fog is composed of several heterogeneous devices, which use different communication protocols. Furthermore, the nodes that make up the fog have several hardware limitations. However, with the knowledge of the limitations of the fog computing environment, the cloud can contribute with its capabilities and resources. Therefore, the process of offloading data from the fog nodes to the cloud makes it possible to use the capabilities of this last paradigm.

The storage in the fog computing paradigm is a topic widely discussed in the literature [3, 11]. It is known that data storage in fog nodes is limited, even with the application of data management techniques and the elimination of unnecessary transactions. Furthermore, this paradigm also applies techniques related to the processing aspect, which aim to reduce the amount of data sent to the cloud [42]. On the other hand, applying these techniques can overload the infrastructure that makes up the fog. It is noteworthy that in the fog, network devices constitute the nodes in the fog, and they are mostly battery powered. Therefore, the application of data compression and processing techniques, while helping to alleviate the restrictions of the fog environment, can overload and worsen the energy autonomy of this paradigm's infrastructure.

In summary, the assessment of infrastructure resources is a requirement that contributes when designing policies for data offloading in the fog. Such resource availability analysis will indicate whether the data flow will proceed normally or whether data offloading will occur [43].

## 4. OffFog Approach

In order to guide the designing of data offloading policies in the context of fog, we propose OffFog. It is an approach that describes essential activities for defining data offloading policies from fog to cloud layers. This approach guides the

Table 1: Example of a security-focused policy.

| Requirements | Objective | Criteria | Techniques |
|---|---|---|---|
| Data | Security | Confidentiality | Authentication and encryption |
| | | Availability | Replication and load balancing |
| | | Integrity | Blockchain |
| | | Privacy | Anonymization |

selection of valuable criteria for data offloading decisions and suggests techniques that support the fulfilment of these criteria in fog computing environments.

Policies play an essential role in the data offloading process. They define how the data will be managed during the offloading process, which security procedures will be necessary, when the process should be executed, which techniques will be used, among other factors. For better understanding, Table 1 presents an example of the basic structure of a data offloading policy. In this example, the policy meets data requirements, has a security-oriented objective, and is based on data offloading criteria given the haze system problems. Finally, the techniques to be used as decision aids in the data offloading process are defined.

Policies are critical for scenarios that require the efficient execution of the data offloading process. The efficient execution of this process brings some benefits to fog computing solutions. Among the benefits, some are remarkable, such as network usage decrease [44], latency reduction [45], energy efficiency [44], and privacy [32, 46]. In this context, a guidance mechanism is required for a taxonomy to be instantiated and used to support policy definition. Therefore, this section presents a methodology to guide the definition of policies for data offloading in fog computing. Therefore, the following activities define actions to be done in the policy design phase.

The purpose of the activities is to ensure that the policy definition accommodates the main offloading criteria classified by the taxonomy. To this end, the guide proposed here seeks to provide an initial methodology that consists of five (5) activities, as presented by Figure 2.

The activity #1 has as the main action understanding the system requirements. The requirement elicitation of the fog system and other software involved is essential to obtain knowledge of the characteristics of the data and the organization's infrastructure. That said, this knowledge serves to support the understanding of the general business requirements and the criteria that influence the offloading process.

As a support for the survey of these requirements, the IoT system requirement engineering support framework proposed by Silva et al. [47] was adopted. The tool was adapted to the fog computing context to consider the requirements of the fog system and the software involved in its solutions.

In this scope, these requirements must be elicited and analyzed to know the system behaviour and the interactions between the components that can help in the offloading decision. In order to provide a tool for this elicitation, we designed a template that shows how this requirements elicitation can be done practically. In summary, based on the Silva et al. [47] method, the suggestion is that the template to identify requirements for fog systems should have an identity field, a description, and a field to define the degree of priority of the requirement under specification.

At the end of the first activity, the architect designing the data offloading policies will have requirements which serve as the foundation for the offloading decisions required by the following activities.

The purpose of activity #2 is to identify the possible data offloading problems. Before defining a policy, it is also necessary to identify the problems that will influence the data offloading process.

There are two situations that must be considered in this activity. The first is the identification of data offloading problems considering that the policy design is for a new system and the second keeping in mind the policy design is for existing systems. For the case of new systems, the problem identification process will be done based on requirements. On the other hand, when the policy design is for existing systems, the problem identification process flow should also consider existing policies and possible problems. In this sense, to assist in identifying possible problems arising from requirements and policies, Table 2 provides a grouping of requirements and relates them to common problems of the fog environment when there is a need for data offloading.

At the end of the second activity, the architect designing the policies will have a list of common problems in fog computing systems. Therefore, the execution of this activity adds to the policy design stage the identification of the main problems that will motivate the definition of the objectives of the data offloading policies.

Activity #3 will serve to define the policy objectives. The definition of these objectives is a central action in the design of policies for data offloading. Therefore, this activity helps the architect to identify the main criteria that may impact the system's cloud data offloading process.

Table 3 guides this identification and organizes the criteria in the data view, i.e., considering the characteristics of the data trafficked in the fog. This support tool will serve to guide the architect to define the system's data offloading criteria and consequently define its objectives. According to Tables 3 and 4 by answering some questions about the system based on its requirements, it is possible to identify which data offloading criteria are related. Therefore, as these criteria refer to a certain objective, at the end of the analysis of the questions, we have two definitions: criteria and objectives.

Similarly, the environment view perspective (Table 4), i.e., helps in the identification of data offloading criteria

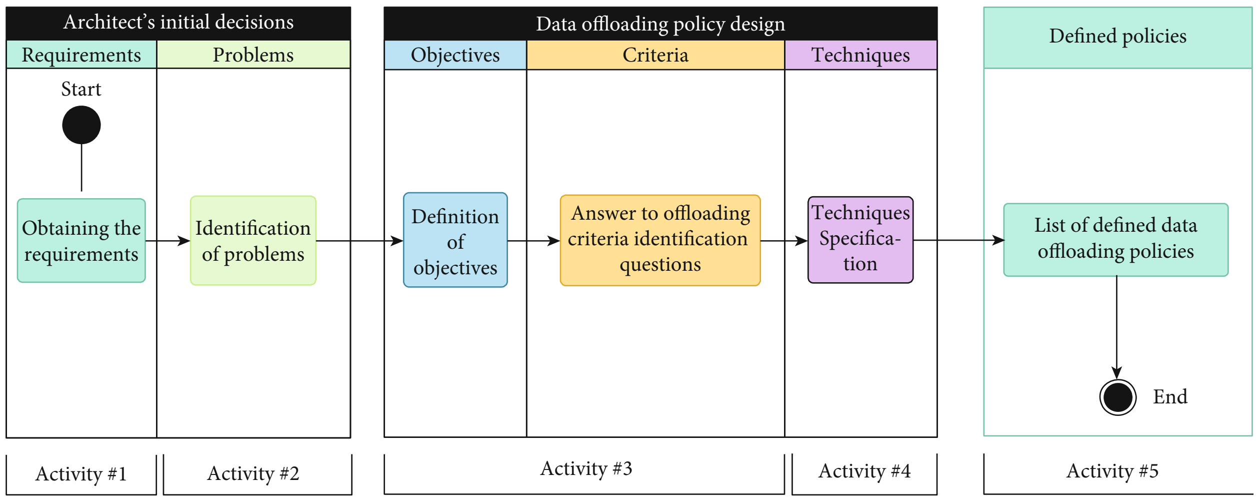

Architect's initial decisions
Requirements
Problems
Start
Obtaining the requirements
Identification of problems
Data offloading policy design
Objectives
Criteria
Techniques
Definition of objectives
Answer to offloading criteria identification questions
Techniques Specification
Defined policies
List of defined data offloading policies
End
Activity #1
Activity #2
Activity #3
Activity #4
Activity #5

FIGURE 2: Activities for defining data offloading policies.

TABLE 2: Relationship between requirements and most common problems.

| Requirements | Problems |
|---|---|
| Data communication | (i) Data redundancy compromises fog node storage [17]<br>(ii) Continuous flow of sensor data compromises storage components [41]<br>(iii) Local data processing compromises processing components [42]<br>(iv) Loss or modification of data [48] |
| Security | (i) Incorrect approaches [49]<br>(ii) Data theft by malicious users [28]<br>(iii) Display of sensitive personal information [50] |
| Availability | (i) Loss of critical data [51] |
| Communication between systems | (i) Network unavailability [52]<br>(ii) Power unavailability [53] |

TABLE 3: Identifying data offloading criteria in the data view.

| ID | Questions | Criteria | Objectives |
|---|---|---|---|
| Q1 | Does the system handle sensitive or confidential data? | Confidentiality | Security |
| Q2 | Is the data accessed frequently? | Availability | |
| Q3 | Does the loss or modification of part of the data cause any harm? | Integrity | |
| Q4 | Should you control data sharing? | Privacy | |
| Q5 | Is the data prefiltered and analyzed beforehand? | Optimization | Management |
| Q6 | Is there data redundancy? | Replication | |
| Q7 | Is there a need for long-term data storage? | Volatility | Nature |
| Q8 | Can data availability tolerate delays? | Criticality | |
| Q9 | Is there a need to reduce the space occupied by data on a particular device? | Size | Volume |
| Q10 | Is there simultaneity between data generation and data storage? | Flow | |

based on the characteristics of the organization's infrastructure. It is important to note that the choice of objectives and criteria for offloading of the policy under development will occur from the favorable response to each question. Thus, each positive answer to the questions in relation to the system requirements will define a criterion to be considered in the policy objective that is being designed in the infrastructure view.

As a result of this activity, based on the answers to the criteria identification questions, the architect will have the definition of their policy objective as a result. Therefore, performing this activity adds to the policy design

Table 4: Identifying data offloading criteria in the environment view.

| ID | Questions | Criteria | Objectives |
|---|---|---|---|
| Q11 | Is the system sensitive to the latency of requests? | Network | Infrastructure |
| Q12 | Does the infrastructure of the scenario have its own storage resources? | Storage | |
| Q13 | Does the system have favorable energy autonomy? | Energy | |
| Q14 | Do the fog devices have good processing power? | Processing | |

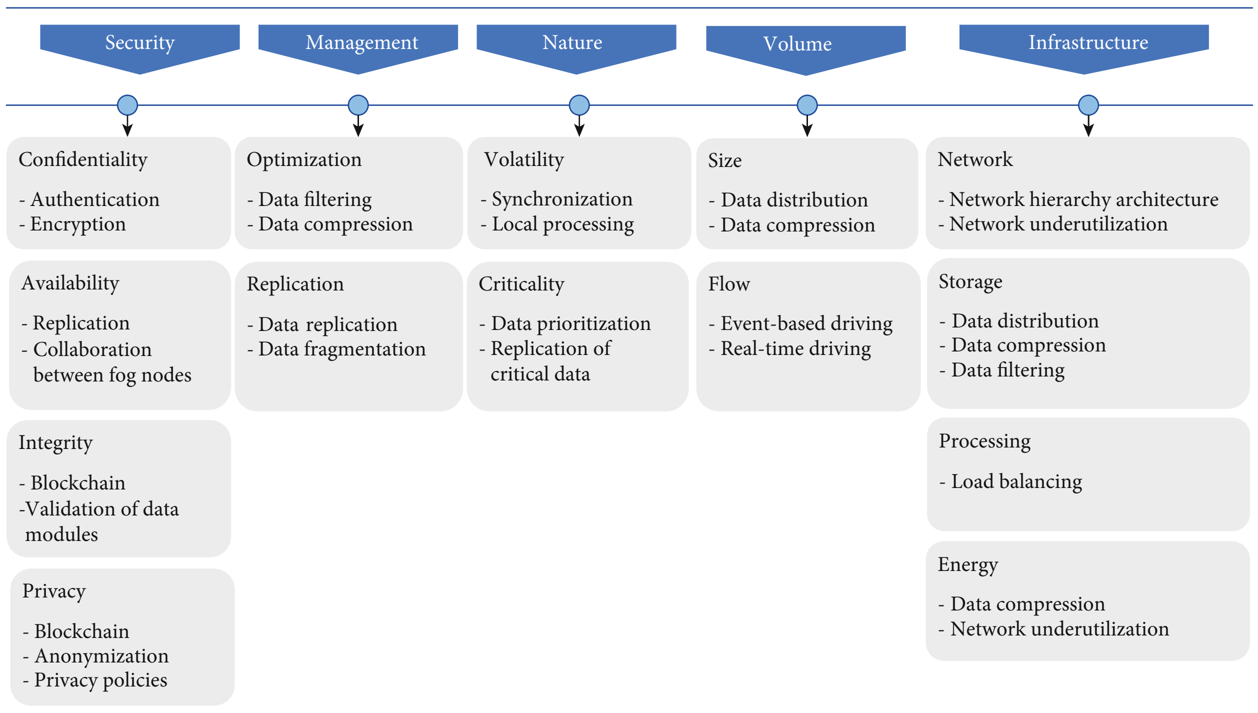


Figure 3: Data offloading criteria and techniques.

phase the ability to establish a relationship between the actual system needs and requirements and the policy objectives.

However, it is still necessary to specify the techniques related to each data offloading criterion for the organized requirements, problems, and criteria to support policy design. With this in mind, the actions of this activity #4 will be focused on selecting methods and techniques to support the data offloading process. This selection will be made based on the highest priority requirements. Therefore, this activity will specify the techniques that will be part of the system's data offloading policies, usually considering the highest priority problems.

Initially, this activity strives to understand the criteria of each policy data offloading objective. With this in mind, the next step is to specify which techniques are essential in executing the fog-cloud data offloading process. Figure 3 simply lists some techniques that were discovered through the study organized by the data offloading taxonomy present in Section 3. By looking at it, it is possible to highlight the most commonly used techniques in each data offloading criterion, which will make it easier to define the actions of the policy being defined.

Activity #5 assists in analyzing the output of the process specified here, that is, the list of policies defined for the solution. The main goal of this activity is to select the conflicting policies. The action defined by this activity is necessary due to the possibility of fog systems having more than one data offloading policy that conflict. In this activity, one must consider that policies can result in different solutions, benefiting or worsening the data offloading in the system. Therefore, it is a crucial decision to be made.

When there are conflicts between policies, the architect should look for alternatives based on his experience and knowledge of the system requirements. Activity #1 can support the decisions regarding requirements, which helps gather requirements based on priority level. So an alternative is to consider this priority level to select which policy is most relevant for the system.

In summary, within the activities required to define data offloading policies, many actions are performed. The first action is formed by the architect's decisions about the requirements and problems addressed in the policies. After these decisions are made, the definition of policies begins. Some questions are raised to identify the criteria that will motivate the data offloading process with the objective

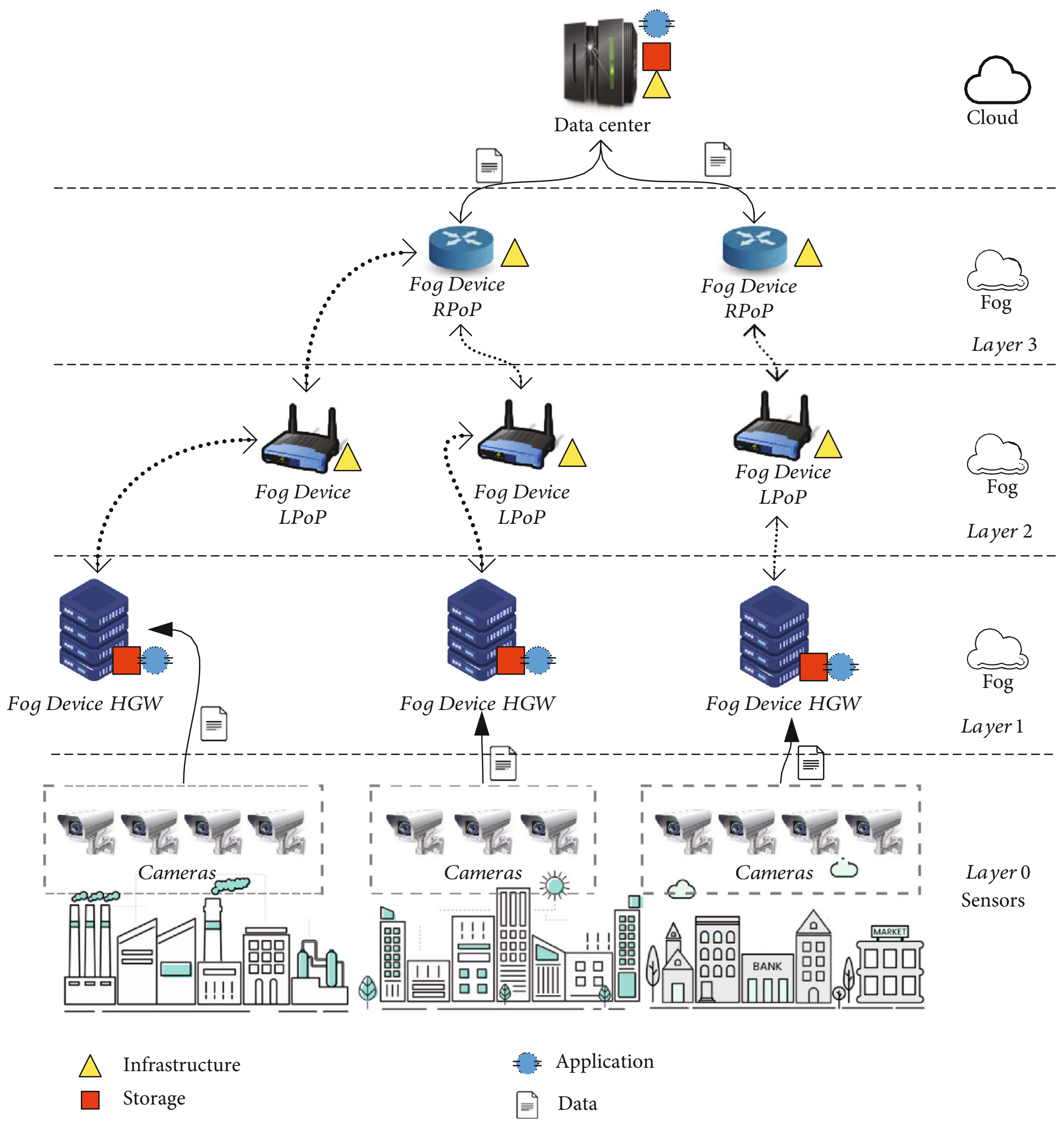


Figure 4: System architecture (adapted from Naas et al. [24]).

defined. Then, after the questions are answered, the techniques used to solve the solution's offloading problem are specified. Finally, there is an output with a policy for each identified problem, which allows decision-making to be performed on conflicts of choice between policies.

## 5. Case Study

In order to demonstrate the practical use of the OffFog approach, we applied it in an urban security application, particularly in a surveillance camera system (SCS). The study is based on the scenario proposed by Gupta et al. [7] but also includes requirements related to high-scale IoT infrastructure for smart cities. This case study attempts to demonstrate the feasibility of using OffFog in fog computing applications.

Intelligent cameras send a detection alert and video stream to a designated controller upon detecting motion in their field of view. As a result, SCS must handle motion detection alerts as quickly as perceived by multiple sensors. Furthermore, a long-term data storage capacity from the systems is required, driven by the nature of several uses for captured images. For that, SCS demands low latency communication to report activities that are currently taking place and also synchronise with other cameras. Therefore, a cloud-based centralised system scenario suffers from a disadvantage because all sensors must forward data to the cloud to handle all processing, causing higher network latency and bandwidth consumption.

For our analysis, we considered a distributed architecture for the SCS, where cameras are responsible for monitoring public places based on the scenario proposed by Hong et al. [54]. Camera sensors detect activities in the context and send the video stream to intermediary devices previously configured. In turn, these devices perform tasks like data compression, partial store, data forward, and data criticality.

Table 5: Data offloading policies for the camera surveillance system.

| Policy ID | Requirements | Objectives | Criteria | Techniques |
|---|---|---|---|---|
| P1 | Long-term image storage | Nature | Volatility | Storing video surveillance images in the cloud |
| | Video stream management for emergency response | | Criticality | Storing data in fog nodes |
| P2 | Real-time data flow management | Volume | Flow | Event-driven data flow for noncritical data |
| | Camera file data size management | | Size | Data compression in fog nodes |
| P3 | Object and motion detection | Infrastructure | Storage | Storing data in fog nodes |
| | Security systems with decision making without delay | | Latency | |

*5.1. Surveillance Camera System Architecture.* Devices LPoP and RPoP organisationally cover infrastructure nodes disposal in the local or regional network. They are responsible for the communication between devices. Data collected by the sensors (cameras) are stored and disposed of on devices installed on fog (HGW) and cloud (DC) layers. Figure 4 shows the arrangement of components present in the SCS architecture.

Intermediate devices HGW are responsible for receiving data captured by sensors or transmitted from another device via LPoP node. Also, they can serve data to system applications. Furthermore, they operate under established data offloading policies. For this, they send data to other HGW devices or to the cloud, according to resource availability, such as storage capacity and data relevance. A set of sensors strictly communicates with a node in the fog set on the topology definition. Sensors are responsible for inserting data obtained and its properties such as size, compression ratio, and relevance in the simulated scenario.

*5.2. Data Offloading Policies for the SCS.* The data offloading policies for the SCS scenario accord to the activities specified by OffFog. Table 5 presents the structure of this design and highlights the three policies defined. According to OffFog guidelines, the policies were organized considering the priorities of the scenario requirements, their objectives, the definition of data offloading criteria, and the specification of techniques that support the offloading decision.

According to Table 5, policy P1 defines that the nature of the data in this scenario should be considered. Data offloading criteria such as data volatility and criticality must be considered to meet two system requirements. The first requirement concerns the need for long-term persistence of the monitoring images. For this requirement, the suggested technique is to store nonvolatile data in the cloud. The second requirement is the management and analysis of the video streams for emergency service response. For this requirement, the suggested technique is to store data in the nearest fog nodes to decrease the latency for decision making in emergencies.

The P2 policy aims to control the volume of data traffic in the system. The system requirements that motivate this policy are the need to manage the data flow in real-time and manage the size of the video files captured by the camera. For the data flow control requirement, the suggested technique is task scheduling (for noncritical data), i.e., data offloading will only occur after some conditional factor, such as when the storage reaches its total capacity. Similarly, the suggested technique is data compression for the data size requirement, aiming to decrease the disk space occupied by the monitoring videos.

Regarding the P3 policy, the focus is to consider the management impact of some resources of the environment infrastructure on the data offloading process. To this end, the first requirement to be considered concerns the storage of object detection and motion data made by the cameras. For this requirement, the suggested technique is to store it in fog nodes near the device, reducing communication latency and allowing the system to recognise objects and their movements faster. Finally, emergency response systems should work without delays that impact their operation, so the suggestion is to prioritise the storage in fog nodes.

## 6. Experimental Analysis

Aiming to evaluate the resulting policies of OffFog adoption on the scenario presented in Section 5, we executed an experimental study based on the well-known framework *iFogSim* [7]. In particular, we provide a fog computing simulation environment to reproduce the SCS behaviour following the resulting offloading policies from the OffFog execution.

Furthermore, we seek to provide insights and also evaluate the trade-offs related to the resulting policies. We evaluated from a latency-sensitive perspective, based on the Naas et al. [24] architecture. We performed several simulations using different values to relevant attributes for the offloading techniques resulting from OffFog (device storage capacity, level of data compression, and data relevance). We offer a topology for this use case, including cloud devices, fog devices, and sensors.

*6.1. Implementation.* Despite the *iFogSim* allows the elaboration of complex and heterogeneous evaluation scenarios, as pointed out by Naas et al. [8], the base framework lacks good data abstraction. The Tuple class represents a data fragment and elementary information, but it does not elaborate on storage abstractions. In their study, Naas et al. [8] introduced improvements to the base framework to evaluate different aspects of data placement, i.e., how to better distribute the data, using different strategies, thus, reducing overall latency.

We extended the *iFogSimWithDataPlacement* (https://github.com/medislam/iFogSimWithDataPlacement) by including new capabilities related to data offloading. Figure 5

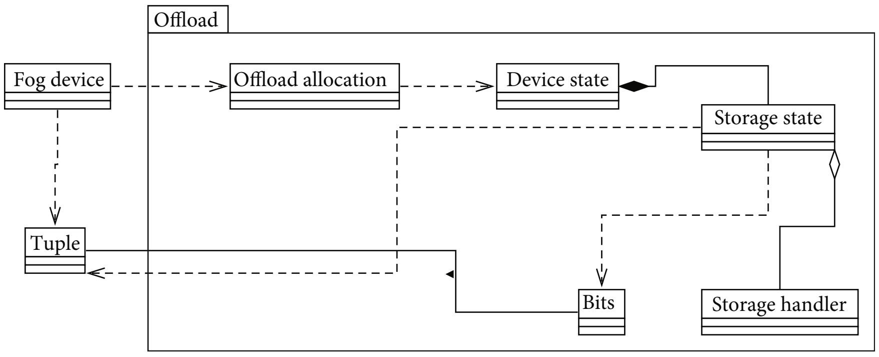


Figure 5: Data offloading implementation components.

Table 6: Data offloading simulation parameters.

| Parameter | Description | Values |
|---|---|---|
| <*DV*>_TUPLE_FILE_SIZE [a] | Data fragment size | 102400 (100 KB) |
| <*DV*>_Storage [a] | Storage capacity | 1048576 (1 MB) |
| <*DV*>_Storage_Min_Threshold [a] | Storage lower-level threshold percentage | 10, 20 |
| <*DV*>_Storage_Max_Threshold [a] | Storage upper-level threshold percentage | 20, 30, 40 |
| <*DV*>_Storage_Compression [a] | Data fragment compress level | 20, 40, 60 |
| <*DV*>_Critical_Selection [a] | Probability of data fragment CRITICAL | 0.3, 0.5, 0.7 |
| <*DV*>_Compression_Selection [a] | Probability of data fragment COMPRESS | 0.3, 0.5, 0.7 |

[a] <*DV*> represents the infrastructure device type (DC, RPoP, LPoP, or HGW).

describes the main components used by our extension. The OffloadAllocation acts as the data offloading entry point, reasoning which host should store the given data. It answers whether it should be kept in the current host or be offloaded to a corresponding node, i.e., to the Cloud (other nodes in the infrastructure may be considered, but it was still not implemented in this version). Other essential components are DeviceState and StorageState, which abstract a device and its current storage state. These components keep track of the stored data and the storage free space, acting as the building blocks to the implementation of data offloading policies. All resources, including source codes, are available at *iFogSimOffload* on Github (https://github.com/labcomu/iFogSimOffload).

Once a new Tuple (data fragment) arrives at OffloadAllocation, it is wrapped with data headers to support the data offloading policies. The Bit component represents these headers. Currently, it provides the options of COMPRESS and CRITICAL. These options indicate if the StorageState should compress the fragment size and if the data fragment should never be offloaded, respectively. These characteristics are selected randomly according to the simulation parameters.

Requests to OffloadAllocation initiate a store attempt into the device storage—performed on StorageState. First, if the COMPRESS bit is activated, it compresses according to the compress level passed as a simulation parameter. Then, it tries to store it, failing when there is no space left on the device. If it succeeds, but the currently used space reaches an upper-level threshold, the store operation initiates data offloading to the corresponding node (for instance, Cloud node). The StorageState starts moving data to the corresponding node, halting when the used space reaches a lower-level threshold. The StorageHandler component handles data offloading and halts logic.

Each simulation takes into consideration parameters that influence the offloading behaviour, as detailed in Table 6. These parameters are configured on the DataPlacement component, which initiates the framework execution. The <*DV*>_TUPLE_FILE_SIZE parameter is closely related to the corresponding <*DV*>_Storage parameter and should be defined in conjunction. Otherwise, the data offloading operation might not be triggered. In other words, if a storage free space is too large compared to the Tuple data fragment size, the used space will remain under the upper threshold—hence, data offloading will not be required. The <*DV*>_Compression_Selection and <*DV*>_Critical_Selection parameters define the probabilities where a data fragment has the COMPRESS and CRITICAL bits selected, respectively. For instance, if we want to select only 20% of data fragments arriving at HGW devices as critical, we should define HGW_Critical_Selection to 0.2.

*6.2. Experimental Setup.* All simulations were performed on an Intel Core i7 4 CPU 1.8 GHz with 16 GB of RAM. Since the *iFogSim* base framework does not benefit from parallel processing and uses only up to 3 GB of RAM, the simulation duration is only influenced by the unitary core frequency (1.8 GHz). Moreover, we defined a topology with 5 DC, 10 RPoP, 50 LPoP, and 500 HGW. Table 6 details the main parameter values varied on simulations. In order to achieve better confidence, we performed three full executions of all possible combinations, producing a total of 405 executions (135 per full simulation) and combined the results by averaging the corresponding values of each configuration. Because we focus on investigating data offloading operations and due to the high number of simulations, we kept the HGW_TUPLE_FILE_SIZE parameter as a fraction of 0.1 of HGW_Storage, and both as low as possible. In this way, we observed more data offloading occurring within practical execution times (see Section 6.3).

*6.3. Evaluation.* We grouped the executions by HGW_Storage_Min_Threshold and HGW_Storage_Max_Threshold, i.e., the storage lower-level and upper-level thresholds, respectively. Section 6.1 explains these thresholds, which determine when the offloading starts and halts, according to a storage usage percentage. The groups were tagged with *T*_<Min>_<Max>, where <Min> and <Max> determine minimal and maximal thresholds for offloading, respectively. For instance, the execution T_10_20 starts offloading when storage usage reaches 20% of its capacity and halts at 10%. This storage threshold would trigger more data offloading than T_20_40, which accepts up to 40% usage and halts at 20%.

In order to understand how the OffFog chosen policies affected the system behaviour, we measured the network latencies under different experiment configurations. In particular, we evaluated the average network latencies, which represent the average time spent, in milliseconds, on all network communications between any pair of devices in the SCS topology (Section 5.1). Having lower latencies means that, on average, the simulations triggered fewer communications.

Figure 6(a) shows the average network latency when we vary the HGW_Storage_Compression and keep both HGW_Compression_Selection and HGW_Critical_Selection fixed in the minimum value (0.3). In all simulations, we can notice a reduction in network latencies with higher compression levels. On the T_10_40 configuration occurs the most considerable variation. It represents an excellent example of how the compression policy, notably the more strong compression techniques, could significantly impact the offloading performance. Also, we can notice that with HGW_Storage_Min_Threshold at 20 (T_20_30 and T_20_40), the latency is significantly lower than the others configuration. It demonstrates the positive influence of more tolerant storage thresholds, where fewer data fragments are offloaded, resulting in fewer network communications.

Figure 6(b) shows the results for HGW_Compression_Selection when we keep fixed the HGW_Storage_Compression and HGW_Critical_Selection to the minimum values (20 and 0.3, respectively). Again, the T_10_40 configuration presents the most considerable variation when HGW_Compression_Selection is between 0.5 and 0.7. It repeats the same pattern observed in Figure 6(a). To some extent, both cases are related to the compression policy. With more data fragments selected for compression, the storage threshold will delay reaching the upper level, resulting in fewer data offloading. Once more, with HGW_Storage_Min_Threshold at 20, the latency is significantly reduced to approximately half due to fewer data offloading.

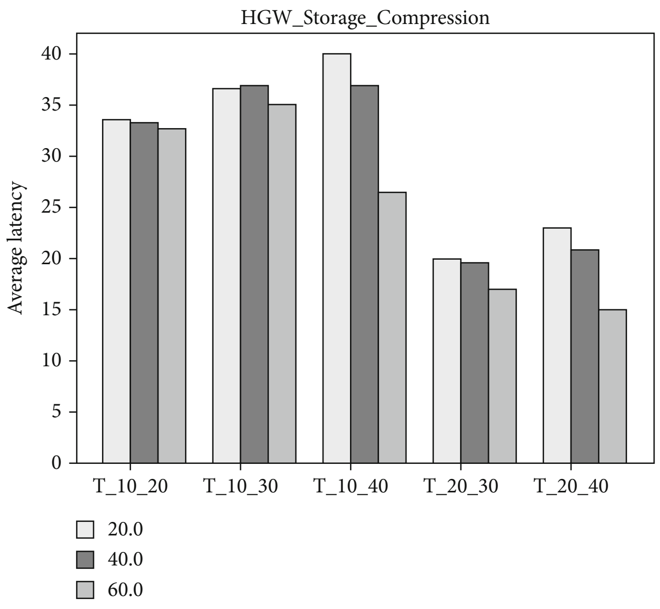
Figure 6(c) shows the results for HGW_Critical_Selection when we keep fixed the HGW_Storage_Compression and HGW_Compression_Selection to the minimum values (20 and 0.3, respectively). Changes in HGW_Critical_Selection produce significant latency variation, especially with values between 0.5 and 0.7. Indeed, the criticality policy directly affects offloading behaviour, given that a data fragment tagged with a CRITICAL bit would never be offloaded. More critical data fragments indicate that a smaller amount of data stored is available for offloading. However, it is relevant to note that this is not always feasible in practice since a critical data fragment will never be sent to the Cloud, which can be undesirable in many real-world scenarios. Finally, we can observe again a reduction in the latency (approximately 50%) when the HGW_Storage_Min_Threshold is the minimum (i.e., 20).

Table 7 presents the evaluation of the results of the two strategies: data offloading (offload, i.e., our extension) and data sent directly to the cloud host (cloud, i.e., without data offloading activated). The two strategy latencies are presented regarding their absolute values in milliseconds (time) and the ratio between offload and cloud (ratio). Data offloading outperforms cloud strategy in all latency statistics by at least 76% (max. ratio 0.24). On average, data offloading reduces the network latencies by 87%. These represent a remarkable improvement, demonstrating the efficacy of selecting significative data offloading policies.

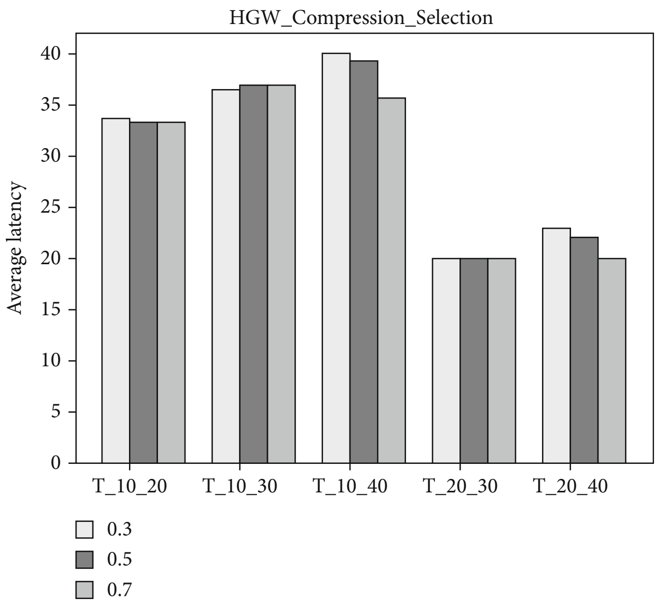
One also can note that the minimum average latency is equal to 1 ms (Table 7). Very low latencies occurred when we simulated HGW_Compression_Selection at 0.7 and HGW_Storage_Compression at 60. It means that with a high number of compressed data fragments and at an elevated compress level, data offloading is reduced drastically. However, such behaviour can compromise data share between devices, which can be undesirable in real-world situations. Therefore, these data offloading policy configurations should be balanced according to the application scenario.

Although the average execution time presented a modest improvement of 5% (Avg. Ration 0.95), the execution was faster in all 405 simulations. With data offloading enabled, less processing occurs when Tuple components (data fragments) are not transferred to other hosts in the simulated topology. It is an interesting behaviour because the same is expected to happen in real-world scenarios.

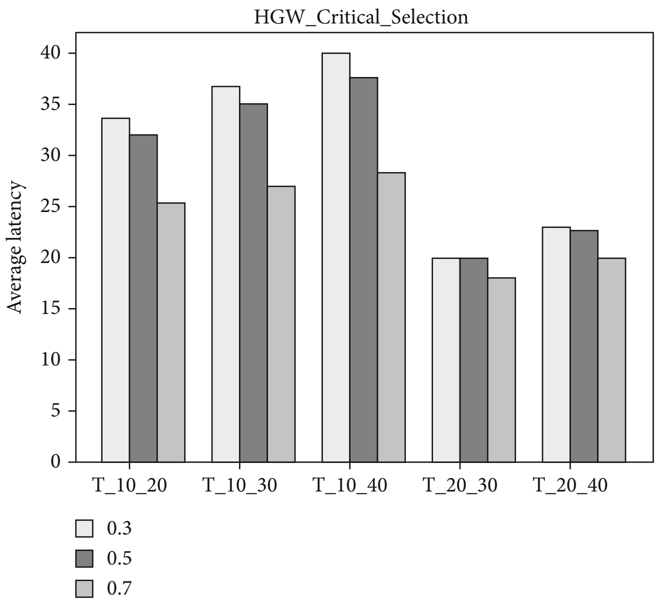
In summary, the experiments with *iFogSimOffload* revealed the OffFog's selected policies effectiveness (Table 5). Grouping results by minimal and maximal thresholds (<*DV*>_Storage_Min_Threshold and <*DV*>_Storage_Max_Threshold) demonstrated a direct influence on network latencies. With more tolerant ranges, fewer data fragments are offloaded. Nevertheless, it should be selected with caution. Otherwise, it might even lead to no data sharing at all. The same considerations apply to data criticality (<*DV*>_Critical_Selection). Simulations with data compression parameters (<*DV*>_Storage_Compression and <*DV*>_Compression_



(a) HGW_Storage_Compression



(b) HGW_Compression_Selection



(c) HGW_Critical_Selection

FIGURE 6: HGW_Storage_Compression latency variation per storage threshold.

TABLE 7: Comparison between cloud and offload statistics.

| Cloud | | Average latency Time (ms) | Average latency Ratio | Execution time Time (ms) | Execution time Ratio |
|---|---|---|---|---|---|
| | | 165 | — | 53,776 | — |
| Offload | Min. | 1 | 0.01 | 48,145 | 0.90 |
| | Max. | 40 | 0.24 | 53,536 | 1.00 |
| | Avg. | 22 | 0.13 | 51,230 | 0.95 |
| | Median | 22 | 0.13 | 51,910 | 0.97 |

Selection) also demonstrate direct influence on latency reduction. Again, it should be considered carefully due to the need for more processing related to compression tasks.

It is important to note that *iFogSim* considers a constant data fragment size. It represents a limitation of our work since we cannot perform more refined analysis with dynamic data fragment sizes. Moreover, because the storage capacity directly influences the time to trigger data offloading, our experiments become unfeasible to execute with large storage sizes. Thus, we kept this configuration more tangible, i.e., 100 KB data fragments and HGW storage capacity of 1 MB.

## 7. Conclusions

Although data offloading is a well-known aspect of fog computing applications, the specification of offloading policies is still an open issue due to the lack of clear guidelines. Therefore, software teams must have knowledge bases arranged in an organized and structured manner to support data offloading decision making for such specifications. OffFog provides this support and, in addition, it also offers a set of activities that guides the policy design from requirements prioritisation to conflicting requirements resolution.

The experiments with *iFogSimOffload* demonstrated the benefits of implementing data offloading based on OffFog recommended policies in an urban surveillance system. Once a real-world scenario topology is implemented, the *iFogSimOffload* can help to identify the best relation between configurations. An optimal combination of compression level, compression selection, critical selection, and offloading thresholds significantly reduces the amount of data offloaded. Indeed, our experiments show an improvement of at least 76% of network latencies compared to cloud strategy. Also, *iFogSimOffload* is executed faster in all simulations, another positive effect of implementing OffFog policies.

In future work, we plan to extend *iFogSimOffload* in order to analyse power-related policies. This feature would allow the inclusion of power-sensitive offloading policies, e.g., data offloading to the cloud during low-battery conditions, thus, avoiding data loss. In particular, we can extend the existing *iFogSim* ability to track device power consumption.

## Data Availability

The data used to support the findings of this study are included in the article. More detailed resources were also publicly published, including source code and simulation results. These are available at the iFogSimOffload on Github repository (https://github.com/labcomu/iFogSimOffload).

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

The authors would like to thank the Department of Informatics and Applied Mathematics (DIMAp) and the Coordination of Superior Level Staff Improvement (CAPES) for the assistance and support provided in all stages of this work. This study was financed, in part, by the Coordination of Superior Level Staff Improvement—Brazil (CAPES)—Finance Code 001.

## References

[1] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–43, 2017.

[2] G. Javadzadeh and A. M. Rahmani, "Fog computing applications in smart cities: a systematic survey," *Wireless Networks*, vol. 26, no. 2, pp. 1433–1457, 2020.

[3] I. Azimi, A. Anzanpour, A. M. Rahmani, P. Liljeberg, and T. Salakoski, "Medical warning system based on internet of things using fog computing," in *2016 International Workshop on Big Data and Information Security (IWBIS)*, pp. 19–24, Jakarta, Indonesia, 2016.

[4] G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli, "A simulation platform for largescale internet of things scenarios in urban environments," in *Proceedings of the First International Conference on IoT in Urban Space*, pp. 50–55, Rome, Italy, 2014.

[5] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[6] S. Melo, C. Silva, and G. Aquino, "Classification aspects of the data offloading process applied to fog computing," in *International Conference on Computational Science and Its Applications*, pp. 340–353, Cagliari, Italy, 2021.

[7] H. Gupta, A. Vahid Dastjerdi, S. 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.

[8] M. Naas, P. 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 2017*, pp. 97–104, Madrid, Spain, 2017.

[9] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: promises and challenges of IoT in medicine and healthcare," *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018.

[10] C. A. Silva, *A Fog Computing-Based Software Architecture for Patient-Centered Management of Medical Records*, PhD thesis, Federal University of Rio Grande do Norte, 2020.

[11] C. A. Silva and G. S. Aquino Jr., "Fog computing in healthcare: a review," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1126–1131, Natal, Brazil, 2018.

[12] C. A. Silva, G. S. Aquino Jr., S. R. M. Melo, and D. J. B. Egídio, "A fog computing-based architecture for medical records management," *Wireless Communications and Mobile Computing*, vol. 2019, 16 pages, 2019.

[13] A. Alelaiwi, "An efficient method of computation offloading in an edge cloud platform," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 58–64, 2019.

[14] F. Pisani, V. Martins do Rosario, and E. Borin, "Fog vs. cloud computing: should i stay or should i go?," *Future Internet*, vol. 11, no. 2, p. 34, 2019.

[15] Y. Shan, H. Wang, Z. Cao, and K. Yury, "Data offloading in heterogeneous dynamic fog computing network: a contextual bandit approach," in *2021 3rd International Conference on Computer Communication and the Internet (ICCCI)*, pp. 73–77, Nagoya, Japan, 2021.

[16] S. He, B. Cheng, H. Wang, X. Xiao, Y. Cao, and J. Chen, "Data security storage model for fog computing in large-scale iot application," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 39–44, Honolulu, HI, USA, 2018.

[17] S. Verma, A. K. Yadav, D. Motwani, R. Raw, and H. K. Singh, "An efficient data replication and load balancing technique for fog computing environment," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 2888–2895, New Delhi, India, 2016.

[18] M. Mukherjee, V. Kumar, S. Kumar et al., "Computation offloading strategy in heterogeneous fog computing with energy and delay constraints," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–5, Dublin, Ireland, 2020.

[19] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Communications*, vol. 14, no. 11, pp. 59–68, 2017.

[20] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[21] M. S. Bali, K. Gupta, D. Koundal, A. Zaguia, S. Mahajan, and A. K. Pandit, "Smart architectural framework for symmetrical data offloading in iot," *Symmetry*, vol. 13, no. 10, p. 1889, 2021.

[22] S. V. Margariti, V. V. Dimakopoulos, and G. Tsoumanis, "Modeling and simulation tools for fog computing—a comprehensive survey from a cost perspective," *Future Internet*, vol. 12, no. 5, p. 89, 2020.

[23] 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 the 10th International Conference on Utility and Cloud Computing*, pp. 47–52, Austin, Texas, USA, 2017.

[24] M. Naas, J. Boukhobza, P. Raipin 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)*, pp. 1–8, Washington, DC, USA, 2018.

[25] C. Puliafito, D. M. Gonçalves, M. M. Lopes et al., "Mobfogsim: simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, article 102062, 2020.

[26] N. Mohan and J. Kangasharju, "Edge-fog cloud: a distributed cloud for internet of things computations," in *2016 Cloudification of the Internet of Things (CIoT)*, pp. 1–6, Paris, France, 2016.

[27] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[28] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, "On security and privacy issues of fog computing supported internet of things environment," in *2015 6th International Conference on the Network of the Future (NOF)*, pp. 1–3, Montreal, QC, Canada, 2015.

[29] C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: architecture, use case, and security and forensic challenges," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 105–111, 2017.

[30] 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*, vol. 154, pp. 22–36, 2019.

[31] V. Moysiadis, P. Sarigiannidis, and I. Moscholios, "Towards distributed data management in fog computing," *Wireless Communications and Mobile Computing*, vol. 2018, 14 pages, 2018.

[32] C. A. Silva, G. S. de Aquino Júnior, and S. R. M. Melo, "A blockchain-based approach for privacy control of patient's medical records in the fog layer," in *Anais Principais do XXV Simpósio Brasileiro de Multimídia e Web*, pp. 133–136, Porto Alegre, RS, Brasil, 2019, https://sol.sbc.org.br/index.php/webmedia/article/view/8011.

[33] 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, Helsinki, Finland, 2012.

[34] A. M. Rahmani, T. N. Gia, B. Negash et al., "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[35] A. Yousefpour, C. Fung, T. Nguyen et al., "All one needs to know about fog computing and related edge computing paradigms: a complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[36] P. H. Kuo, A. Mourad, C. Lu et al., "An integrated edge and fog system for future communication networks," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 338–343, Barcelona, Spain, 2018.

[37] S. Esteves, J. Silva, and L. Veiga, "Quality-of-service for consistency of data geo-replication in cloud computing," in *European Conference on Parallel Processing*, pp. 285–297, Rhodes Island, Greece, 2012.

[38] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: a taxonomy, survey and future directions," in *Internet of Everything*, pp. 103–130, Springer, 2018.

[39] H. Shi, N. Chen, and R. Deters, "Combining mobile and fog computing: using coap to link mobile device clouds with fog computing," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pp. 564–571, Sydney, NSW, Australia, 2015.

[40] W. Lee, K. Nam, H.-G. Roh, and S.-H. Kim, "A gateway based fog computing architecture for wireless sensors and actuator networks," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 210–213, Pyeong-Chang, Korea (South), 2016.

[41] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, "Developing iot applications in the fog: a distributed dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, pp. 155–162, Seoul, Korea (South), 2015.

[42] M. Maksimovic, "Improving computing issues in internet of things driven e-health systems," in *Proceedings of the International Conference for Young Researchers in Informatics, Mathematics and Engineering'17*, pp. 14–17, Kaunas, Lithuania, 2017.

[43] N. I. M. Enzai and M. Tang, "A taxonomy of computation offloading in mobile cloud computing," in *2014 2nd IEEE*

*international conference on mobile cloud computing, Services, and Engineering*, pp. 19–28, Oxford, UK, 2014.

[44] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, pp. 89–100, 2019.

[45] 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 (Per Com Workshops)*, pp. 1–4, Sydney, NSW, Australia, 2016.

[46] F. A. Gomes, W. Viana, L. S. Rocha, and F. Trinta, "A contextual data offloading service with privacy support," in *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*, pp. 23–30, Teresina, Piauí, Brazil, 2016.

[47] D. V. Silva, T. G. Gonçalves, and G. H. Travassos, "A technology to support the building of requirements documents for iot software systems," in *19th Brazilian Symposium on Software Quality*, pp. 1–10, São Luís, Brazil, 2020.

[48] A. Alazeb and B. Panda, "Ensuring data integrity in fog computing based health-care systems," in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pp. 63–77, Springer, 2019.

[49] M. Mukherjee, R. Matam, L. Shu et al., "Security and privacy in fog computing: challenges," *IEEE Access*, vol. 5, pp. 19293–19304, 2017.

[50] A. Heidari, M. A. Jabraeil Jamali, N. Jafari Navimipour, and S. Akbarpour, "Internet of things offloading: ongoing issues, opportunities, and future challenges," *International Journal of Communication Systems*, vol. 33, no. 14, 2020.

[51] U. O. Delaware, *Understanding Data Criticality*, 2018, http://www1.udel.edu/security/data/criticality.html.

[52] Z. Hao, E. Novak, S. Yi, and Q. Li, "Challenges and software architecture for fog computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 44–53, 2017.

[53] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, and K. Makodiya, "Fog data: enhancing telehealth big data through fog computing," in *Proceedings of the ASE bigdata & socialinformatics 2015*, pp. 1–6, Kaohsiung, Taiwan, 2015.

[54] 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, Hong Kong, China, 2013.