Effective Strategy Based on Migration and Replication Techniques for Service Management in Fog Environment

Khawla Bekkouche[1], Said Brahimi[2], Boulaiche Mehdi[3]

[1]Computer and Communication Laboratory (LICUS), the University of Skikda, Algeria, Computer Sciences Department. E-mailkh.bekkouche@univ-skikda.dz

[2]Guelma Automation and Computer Laboratory (LAIG), the University of Guelma, Algeria, Computer Sciences Department. E-mail: brahimi.said@univ-guelma.dz

[3]Computer and Communication Laboratory (LICUS), the University of Skikda, Algeria, Computer Sciences Department. . E-mail: boulaiche.mehdi@yahoo.fr

*Corresponding author:kh.bekkouche@univ-skikda.dz

Received: 13/06/2023; Accepted: 20/11/2023

Abstract

Fog computing has been developed to meet the evolving demands of Internet of Things (IoT) applications. However, in the intricate landscape of distributed and geographically diverse IoT environments, the inadequacy of service management presents challenges such as reduced user accessibility and compromised service quality, impacting response times, network congestion, and energy consumption. This paper introduces RM-SAPCC, an innovative service management approach tailored for fog computing. The method addresses the variability of Fog components and the widespread geographical distribution of infrastructures. To achieve this, a migration and replication strategy based on service popularity is proposed. This dynamic approach optimizes query response times and minimizes system energy consumption without overloading the network. Additionally, a process for eliminating redundant replicas is presented, ensuring the proximity of remaining replicas to incoming requests. The effectiveness of the approach is evaluated through simulations using IFOGSIM, with results indicating promise.

Keywords: Index Terms—Migration, Replica, query, placement, IFOGSIM.

TobRegul Sci.™ 2023;9(2):1531 - 1549

DOI: doi.org/10.18001/TRS.9.2.95

1. Introduction

The Internet of Things (IoT) is an infrastructure that connects a wide range of everyday devices. It enables the monitoring of various environmental parameters, such as temperature and presence, through detectors, while also allowing control of the environment through actuators [1]. Over the past few years, the number of connected devices has continued to surge, leading to the generation of massive amounts of service. Typically, IoT services are processed and stored in

the cloud, which proves sufficient for the vast majority of IoT applications in terms of providing ubiquitous access, processing performance, storage capacity, scalability, and availability [2]. However, the cloud is a centralized paradigm. Indeed, the cloud creates bottlenecks due to the increase in the number of connected objects. So, sending all services to these server clouds causes high and unpredictable latencies, and therefore, degradation of the quality of service (QoS) [3]. Additionally, there are many IoT applications, such as telemedicine, that require geo-distributed deployment, mobility support, real-time response, and localization awareness that cloud computing cannot effectively deliver. Fog computing has emerged to remedy these problems [4].
The fog plays the role of an intermediary between end-users and the cloud for reinforcement, utilizing the resources available in the network equipment [5]. Fog Computing is characterized by proximity to users, dense geographical distribution, and support for mobility [3]. The specific features of Fog Computing lead to reconsidering many issues. For example, since service localization is a crucial parameter for processing, the study of how services should be organized in such an infrastructure to reduce latency and increase the availability of services is very important. That is why we are working toward that objective. In our work, we suggest a service management solution through which users can access the service from any location in a faster manner, taking into consideration network load and energy consumption.

Numerous studies in the literature have addressed the theme of managing service in a fog computing environment. Huang et al. [6] and Aral et al. [7] have focused on the placement of many replicas, while Naas et al. [8] have investigated the issue of placing a single replica of the service. Their goal was to lower the overall latency system.
As far as we are aware, there is no one-size-fits-all effective solution for managing latency, network load and energy consumption at the same time.

In this work, we propose a service management method in a Fog environment, which we call ‘RMSAPCC' (i.e. Replication and Migration Based on Service accessibility Prediction with Connectivity based on Clusters). RMSAPCC aims to enhance system performance by reducing power usage and response time. The contribution suggested in this study deals with the replication and migration of services. The following is a summary of this contribution: We suggest a diagram that will carry out the user query. It directs the query to the closest node that has the user's requested service, which is ultimately a connected object. To maintain the service close to the users, we suggest a service replication and migration plan. Our plan is based on an examination of service access history. It is divided into essential stages: the replication and migration stage, it permits the identification of the service that should be replicated or moved in the system. Also, it makes it possible to determine the number of replicas and the clusters that are affected by replication. The replica placement stage enables the optimal sites for service replicas to be chosen within an area. It is based on formulas that have been offered, including the fog index, the weight of clusters, and the value of a service. Finally the stage of deletion redundant replicas in each cluster that allows to meet large number of query without cluttering clusters.

We used the IoT and fog simulation tool iFogSim to put our approach into practice. In fact, the experimental study demonstrated that the outcomes are generally extremely positive.
The rest of this paper is organized as follows: In Section 2, we present some related works existing in the literature. Then, we detail our contributions in Section 3 and 4. Section 5 shows the different results of the experimentation, and we will conclude in Section 6.

Ii.Related Work

This article focuses on the problem of managing service within a Cloud and Fog infrastructure. The aim is to minimize response time without overloading the network and without suffering any energetic consequences. As the issue we are dealing with is recent, we start this section by describing the principal existing works that utilize the paradigm of Fog Computing and the placement of IoT service instances in the fog infrastructure. In fact, these works are closer to our problem in terms of the context of the work (such as fog computing or IoT) and goals (e.g., reducing reaction time and system latency) or simulation Environment used (e.g., iFogSim, CloudSim) Nans et al. in [8], have suggested an extension of a Fog and IoT environment simulator (iFogSim). The purpose of this extension is to produce a platform dedicated to the evaluation of strategies for placing data in the fog. The authors added three elements to the iFogSim simulator. These elements are described as follows:

•Data Placement: This is the main component of this plugin. It provides users with a set of functions to calculate a data placement using methods based on linear programming.

•Infrastructure partitioning: To reduce the complexity of the data placement problem, the authors added infrastructure partitioning features. This component divides the simulated infrastructure into multiple parts and performs a data placement sub-problem in each part of the infrastructure.

•Data Flow Distribution: This component generates different data flow distributions allowing evaluating placement strategies using different data distributions (zoned, distributed and mixed) in a generic “smart city” scenario. However, unique data placement strategies can result in high system latency.

By supplementing the previous work [8] of Nans et al. which deals with the topic of placing a unique copy of the data. Huang et al. In [6] treated the placement of many data replicas in a Fog Computing infrastructure with the iFogStorM strategy. By taking into account the two problems of replication: the number of replicas and the location. Moreover, given the iFogStorM complexity, the authors proposed a MultiCopyStorage heuristic approach that uses a “greedy” algorithm to choose a solution of the data placement problem in a large-scale infrastructure. This algorithm efficiently places data with minimal overall latency. The results of the experiments show that MultiCopyStorage offers good solutions for storing real-time data in the Fog environment. However, such a strategy can be strengthened by taking a greater number of factors into account when formulating the problem, rather than considering storage capacity as the unique limiting factor.

In the article [7], the authors suggested a distributed service placement algorithm that relies on the dynamic creation/replacement/deletion of replicas guided by the continued monitoring of network edge node service requests. In this algorithm, the storage nodes which host the replicas analyze the request observed on the replicas and behave as local optimizers. The authors have treated this problem as a Facility Location Problem (FLP-based combinatorial optimization problem), the nodes evaluate the replication cost of storage as well as the predicted latency to decide whether to migrate or replicate to one of the neighbours in order to improve the objective function of the issue. They can choose whether to delete a local replica. The algorithm also enables the user to control the balance between optimizing costs and optimizing latency. In addition, the authors have added a replica discovery method where concerned nodes are notified of nearby replicas. Experimental results demonstrate that this distributed placement algorithm replicas gives significant advantages in terms of cost and latency, compared to a replication-free approach, as well as client-side data caching that is widely used in traditional distributed systems.

Vales et al. [9] suggested a hybrid hierarchical cloud storage system. The suggested file system centralized the policy and data management and enabled the storage of data and dissemination in a distributed manner. In addition, the authors devise a replication strategy based on the location of nodes defined by the distance between consumers and the popularity of space-time data based on the frequency of data access. The results of the assessment show that this proposal improves the service life of the network, provides data to consumers with low latency, reduces main traffic and allows for equitable power consumption between battery-powered nodes.

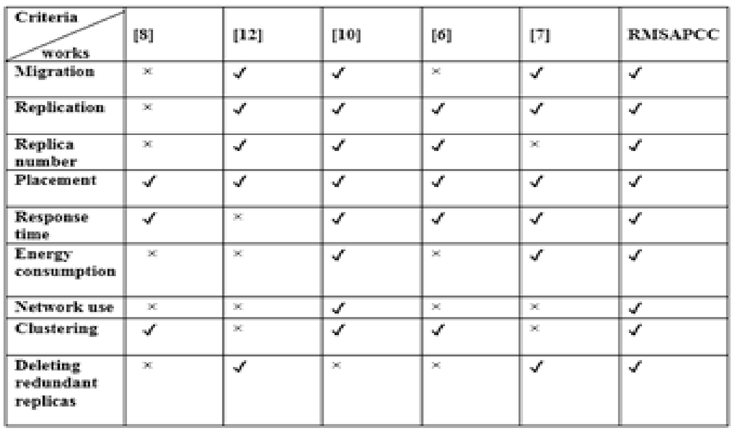
In the literature, a number of studies deal with the issue of the placement of IoT modules in the Fog. This was done with the following goals: (Reduce latency, reduce energy consumption, or improve quality of service). For instance, in [10] the authors suggested an algorithm that allow the placement of instance services of an IoT application in Fog-Cloud infrastructure. This algorithm sorts nodes and service instances according to their capabilities in descending order, then performs iterations between Fog nodes, Cloud nodes and places modules on the nodes.

Benamer et al. in [11] proposed a model that relies on linear programming to maximize the placement of service instances in a fog infrastructure. That is an exact solution, but one of exponential complexity. The authors defined another heuristic solution that aims to find the best placement for each module by taking into account the inter-node latency, the global latency of the system, as well as the capabilities of the Fog nodes. The valuations show a significant decrease in latency compared to both iFogSim placement strategies (Edge Ward and Cloud).

In [10], the authors Berkennou and AL suggested a Fog and Cloud type hierarchical architecture. In which the Fog layer is divided into areas, each area with a catalog of data location. They also suggested an approach for balancing workload by routing requests to the nearest node housing the data requested by a user. In addition; they presented the RMS - HaFC migration and replication approach to minimize the average time and optimize the total energy consumption In the Fog Computing environment. This technique takes into account the geographic infrastructure of Fog as well as the popularity of data. In fact, it specifies the quantity of replicas and their ideal positions in each region. In this approach they assumed that Fog nodes had sufficient resources to host data and handle user requests. This approach remains the best among the other existing approaches if taking the response time, the energy consumption, and the network overload according to authors of [10]. However, to propose a strategy allowing to delete the replicas seems necessary.

In our previous work [12], we have talked about replica migration in fog computing to improve service accessibility. We have proposed two replica migration algorithms that consider service access history called respectively RMSAH (Replication and migration based on service accessibility history) and RMSAHC (Replication and Migration based on the history of service accessibility and contiguous fog nodes). A fog node migrates copies that we expect will have high accessibility in the RMSAH heuristic. In the RMSAHC heuristic, replicas are first migrated using the RM-SAH method, and then replica duplication is removed from all neighbouring fog nodes, allowing other copies to be placed in the desired location. The RMSAHC approach provides the highest accessibility, while the RMSAH method provides the least, according to the simulation results.

Table 1: Our strategy in perspective with some related works.



Criteria / works	[8]	[12]	[10]	[6]	[7]	RMSAPCC
Migration	×	✓	✓	×	✓	✓
Replication	×	✓	✓	✓	✓	✓
Replica number	×	✓	✓	✓	×	✓
Placement	✓	✓	✓	✓	✓	✓
Response time	✓	×	✓	✓	✓	✓
Energy consumption	×	×	✓	×	✓	✓
Network use	×	×	✓	×	×	✓
Clustering	✓	×	✓	✓	×	✓
Deleting redundant replicas	×	✓	×	×	✓	✓

In Table 1, we have tried to position our approach to the existing associated works identified in this section. We have carried out a comparative study of our method which is based on different criteria such as the energy consumption and response time and the various techniques use.

III. Background

Our objective is to propose an approach for the management of the services by using the techniques of replication or migration based on the history of access to the services to guarantee the quality of service (response time, consumption of energy...). This strategy is based on the analysis of service access history. And it is structured in three main steps as shown in figure 1.

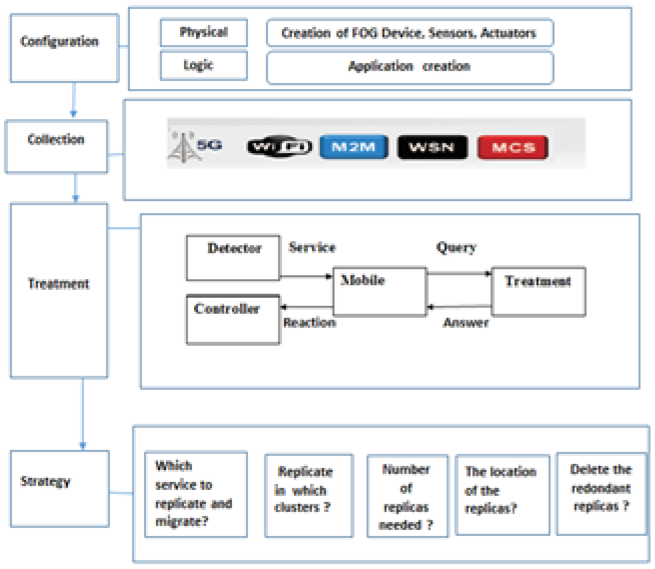

Configuration
Physical
Creation of FOG Device, Sensors, Actuators
Logic
Application creation
Collection
5G
WiFi
M2M
WSN
MCS
Treatment
Detector
Service
Query
Mobile
Treatment
Controller
Reaction
Answer
Strategy
Which service to replicate and migrate?
Replicate in which clusters ?
Number of replicas needed ?
The location of the replicas?
Delete the redondant replicas ?

Figure 1: The stages of our proposal.

• Configuration: in this step we build a Cloud and Fog type platform. In addition, we build the physical and logical elements that will be deployed in the infrastructure.

• Treatment: Running multiple instances of a generic IoT application model.

• Strategy: this step answers the questions about the migration and replication of services such as the number of replicas and so on.

We implemented a generic IoT application model (see Figure 1: treatment). In this case; the connected device receives service from the detector. Then, it requests query from the gateway. This query requires the execution of a certain service; it will be forwarded to the closest host in order to obtain this information. We go into further depth about this procedure in the following section.

A. Running a query

Among the major steps in our proposal, we cite the routing of query to the node that hosts the service requested by a user.

To understand the execution of a query, we used the activity diagram shown in Figure 2.

This activity diagram is used to model a process; it provides a detailed view of the information flows that are exchanged between actors. We present below a detailed explanation of this diagram:

When a query arrives, the node first tests whether it does not need a service, if so, it executes the query. Otherwise, the node launches a search for the requested service. In this situation, three (03) cases may arise:

• If it finds this service locally, it executes the query.

• If it finds the service but not locally, it sends the query to the node having the service.

•If the service is not found locally nor in the node having the service, it sends the query to the root (that has a broad perspective without specifics on every cluster (hosted service, load . . .).

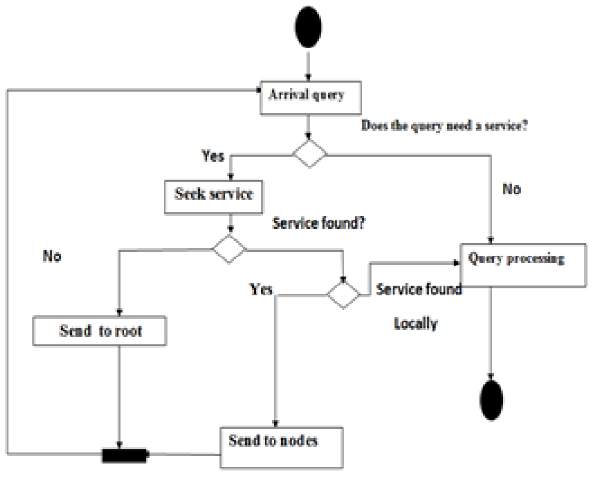


Figure 2: Activity diagram of the arrival of a query.

Iv.Proposed Strategy Rmsapcc

The objective of our article is to propose a large-scale Fog service management system to reduce time and energy consumption while taking into consideration the characteristics of the fog. To do this, we have proposed a service replication or migration strategy called "RMSAPCC (Replication and Migration Based on Service accessibility Prediction with Connectivity based on Clusters)", in order to keep services close to users.

In order to formulate and model our service management strategy in a Fog infrastructure, we relied on a set of assumptions defined as follows.

• A server represents a small data center consisting of multiple hosts in the Fog environment. As opposed to our work, we assume that a server contains only one host. Moreover, a server may host only a single copy of service.

• Consider that our system subdivided into C clusters, each cluster groups together a set of fog node F, moreover we assumed that all the services requested by the users exist in the system.

• We establish an infrastructure with the suggested system. Then, the services will be created, placed arbitrarily and fairly in the infrastructure by a component called Organizer. We assume that the Organizer has a full vision of the complete infrastructure. After a processing step, the Organizer employs our approach which consists of two crucial stages to accomplish the desired goals, each stage goes through a number of sequential processes. The stages that were treated are described below.

A. The replication and migration stage

The replication and migration stage enables to identify precisely which services in the system need to be replicated and migrated. As our infrastructure is subdivided into several clusters, this stage identifies the clusters that are affected by replication while specifying the number of replicas to be created. To address these issues, theaccess history of each service is examined (popularity). The advantage of such a technique is to determine which recently requested services in such a cluster that will be more likely to be requested again in the near future. We detail the steps of this stage below.

(1) The popularity

The popularity of a service (value) measures the quantity of requests made by customers on a service. It represents crucial information, as it gives an indication of the value of the services. This popularity is based on important factors. These factors are the file access frequency, average requests per cluster and time to last request respectively.

The formula for calculating $p(S_i)$ for a service S_i is as follows:

$$p(S_i) = \frac{AAS_i}{NCRS} + \frac{NCS_i}{AT - SLRT} + \frac{NS_iA}{NAAS} \quad (1)$$

Where:

AAS_i: Amount of Access to the Service

NCS: Number of Copies of the Service

NCRS: Number of Clusters Requesting the Service

AT: Actual Time

SLRT: Service Last Request Time

NSA: Number of Service Accesses

NAAS: Number of Accesses of All Services S

Our method takes geographic partition into consideration. Consequently, it is feasible to identify clusters that have a priority for having service copies by looking at a cluster's weight (the predicted candidate clusters to host replicas). The formula is calculated as follows:

$$WC_k = NRS_i C_k + \frac{1}{AT - \frac{\sum_{n_K}^{j=1} TLRS_i S}{n_K}} + \frac{NFC_k}{TNQEC_k} \quad (2)$$

Where:

NRS i CK: The number of requests for services by cluster user CK

AT: Actual time

NFCK: Number of fog node in CK

TNQE CK: Total number of queries executed in CK

TLR Si S: The time of the last query of Si by the server

nK: The number of nodes in the CK cluster

The factor $\frac{NFC_k}{TNQEC_k}$, indicates the reverse of a cluster load, also it is used to favour clusters with minimum loads.

(2) Number of replicas

In this step, we will measure how many replicas must be produced for each service in each cluster. The formula is calculated as follows:

$$X = \frac{AS_i\, RC}{VS_1} \quad (3)$$

ASiRC: The Amount of Service Requests made by Cluster users

VS i: The Value of a Service (popularity).

The report $\frac{AS_i\, RC}{VS_1}$ ensures a minimum number of replicas that meets the potential need of a C cluster. We modelled the step of migration and replication using an activity diagram (shown in Figure 3) to represent the behaviour of our technique and how it unfolds. The diagram shows the processing performed before making the decision to migrate or replicate a service.

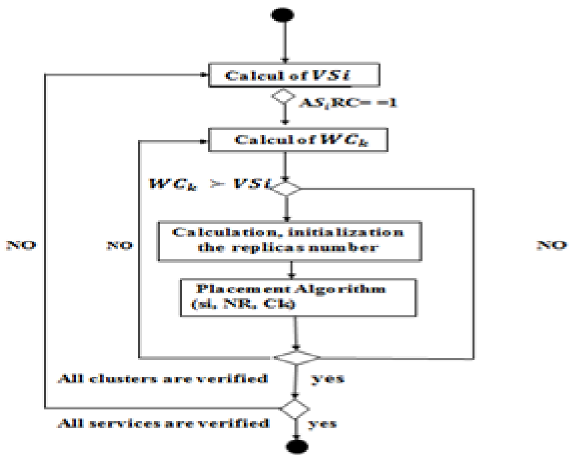

Calcul of VSi
$AS_iRC = -1$
Calcul of WC_k
$WC_k > VSi$
Calculation, initialization the replicas number
Placement Algorithm (si, NR, Ck)
NO
NO
NO
All clusters are verified
yes
All services are verified
yes

Figure 3: Diagram of replication and migration activity.

B.The placement of replicas stage

In this stage, we place the service replicas efficiently. In order to accomplish our goals, such as load balancing and a faster response time.

In fact, the placement algorithm, presented below, is called by the procedure of migration and replication. With that in mind, this step is therefore necessary for the previous stage (migration and replication).

In this stage, the replica placement algorithm calculates the index of each node in the cluster with formula (4). Then it sorts the nodes based on their Index in Highest to lowered order and selects the fog nodes with the maximum index to place the replicas.

$$y = \frac{NSQF_{kj}}{\frac{NQC_k}{NFC_k} * NS_iF} + \frac{1}{AT - TLQS_i} \quad (4)$$

Algorithm: placement

```
Input : service Si, X : Replica number
Output: placement Replica
Begin
  For each FogNode F belong to C do
      Calculate the index Y of FogNode
  End
      SortFogNode based on their index in Highest to lowest order;
      SortedSet= SetofSorted FogNode;

  For i to X do
      Replica =create a service replica
      Place the replica on the FogNode SortedSet[i]
  End
End
```

Where:

Y: Fog Node index

NSQFk j : The number of service requests Si by node Fk j

NQCK: The number of queries in CK

NFCK: the number of fog nodes in CK

N Si F: The number of existing services in the fog node

TLQSi : The time of the last service query.

The Fog Node Index allows evaluating the fog nodes to decide which the optimal node to host the replica is. It is depended on factors like the number of services in order to have good storage. The number of service queries per fog node as well as the last query time in order to favor fog nodes that will likely need service in the future. Figure 4 (placement diagram (, shows this process in the form of an activity diagram.

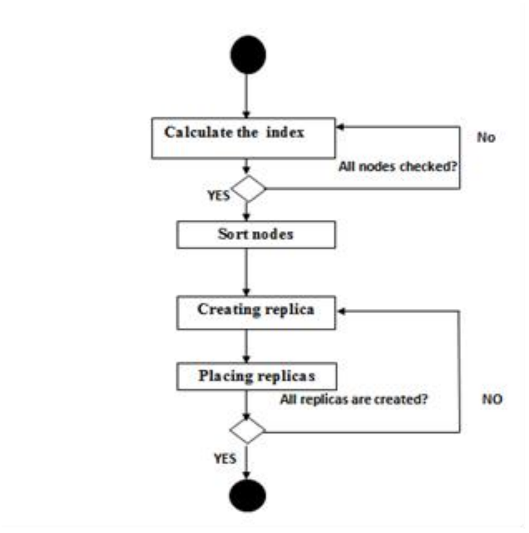


Figure 4: Replica placement activity diagram.

C. Deletion of redundant replicas in each cluster:

•First, each fog node diffuses its identifier as well as information based on popularity. After all fog nodes have completed their diffusion, each node will determine the linked fog nodes based on the received node identifiers.

• The RMSAPCC method eliminates redundancy replicas in larger clusters of fog nodes. So that each cluster of fog nodes cooperates for hosting the largest number of services to meet the query of a large number of users and then get rid of duplication replicas

•When a service item (original/replica) is replicated among the fog nodes in the cluster, the node that holds the replica requests the migration of another replica.

•If there are many replicas among nodes, the fog with the lower popularity changes the replica to another replica (delete the previous replica).

•Replicas of services are cached in the order of the popularity until all fog nodes in the memory space of the clusters is complete.

• Replicas of services that fog node hold as originals are not cached here. Each replica is stored on a fog node with the highest access frequency to the service among nodes with enough free memory.

V. Simulations And Result Analysis

1. Evaluation criteria

In the context of fog computing, the following simulation criteria have the following meanings:

Response Time: This is the overall amount of time needed to process a query in the Fog Computing environment. It includes the service transmission time from IoT devices to the Fog Node, the time of processing at the Fog Node, and the response back time to the device from another device. Better Fog Computing efficiency is indicated by a brief response time.

Energy consumption: This concerns the amount of energy consumed by the fog nodes during the execution of migration and replication tasks. High energy consumption can lead to increase energy costs. It must be said that the lower the energy consumption, the more the approach is environmental.

Network congestion: Defines network overload or saturation in a fog environment. When the number of communications between fog nodes caused by service migration increases, there may be network saturation, which leads to delays in service processing and lower system performance. The more the migration number increases the more the network saturates.

2. Experiment result

To meet the objectives of our work, we extended the iFogSim as shown in Figure 5. This Toolkit is made up of numerous classes that we can categorize as follows:

(i) A set of components that allow the physical equipment of the infrastructure to be modelled.

(ii) A set of components that allow the logical parts of the simulated scenario to be modelled (for example the instances of IoT services and their dependencies).

(iii) A set of components that allow the management of the processing in fog nodes. These resources are controlled to improve service latency, network traffic, power consumption, and system running costs by placing and scheduling serviceinstances in physical components.

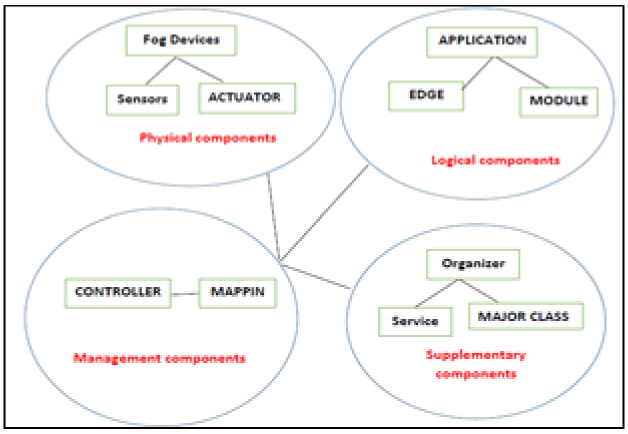


Figure 5: Design of IFOGSIM Extension

(iv) A set of components, that we have created, model the principal parts of our contribution, such as configuration andservice management.

The classes below represent the logical aspects of our contribution (for example, configuration and service management), as indicated in the figure 5.

• MAJOR CLASS: This is the core class, and it provides a collection of options for configuring a Fog platform. The logical aspects of the simulated scenario that will be deployed in the infrastructure are likewise created using this class.

• ORGANIZER CLASS: This class has a comprehensive view of the infrastructure. It specifies a collection of service management strategies for use in the simulated infrastructure. We can also use this class to start our replication and migration approach.

• SERVICE CLASS: (i.e., service deployed in the simulated infrastructure.)

It specifies a set of attributes such as its name (for example, service 1), size (in bytes), fog node name hosting the service, and it is type (i.e., original or replica). In each node there is an array of services. The units (sensor, fog devices, and actuator) in the iFogSim Toolkit communicate with one another using preset events. There are events for launching a service instance in a Fog node, connecting a sensor, and sending service to another component, for example.

It is worth noting that we have simulating the extra events for specific situations. To highlight the participation of our methods, we will focus on the three metrics :(response time, energy consumption and network congestion). In order to evaluate the behaviour of our three propositions and to discuss its results obtained through simulation we will compare them.

A series of simulation were launched by the following several parameters shown in Table 2.

Table 2: Simulation parameters

Parameter	Value
Cloud latency	200
Server bandwidth	8000 Mbps
Number of fog nodes by clusters	5
Clusters number	4
Number of queries	[300, 500, 700, 900]
Number of service	100
Service size	200 KO

A. Average response time

The average response time obtained with the methods applied is measured in this experiment for all queries in the simulation. The formula is as follows:

$$\text{Average response time} = \frac{\textbf{TNQ}}{T} \quad (5)$$

Where:

The total number of queries is represented by TNQ.

T: denotes the amount of time that has passed between the user sending "i" query and receiving a response from the required service.

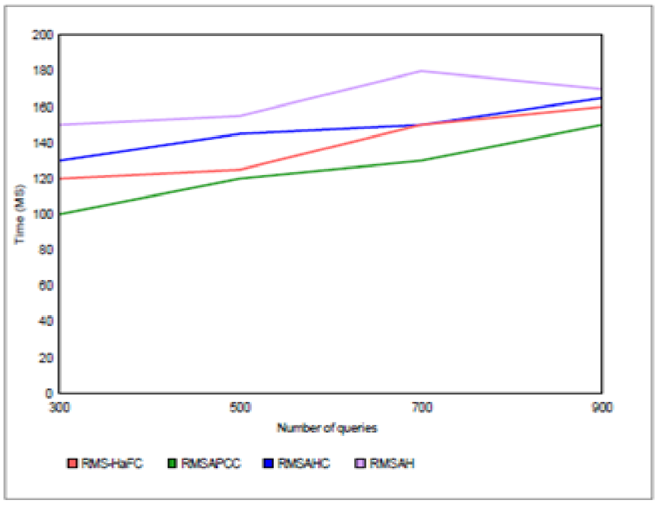


Figure 6: Response Time metric.

Figure 6 shows a considerable reduction in response time when we use RMSAHCC technique, which is due to the replication mechanism we used to enhance response time and which practically eliminates duplication while allowing a high number of services to be deployed near of users.

B.Energy consumption

Energy consumption depends on several factors such as processing power, standby power consumption, and specific node operating parameters. The formula of energy consumption is calculated as follow:

$E = SPC + DPCR \times NQ$ (6)

Where:

SPC: Static power consumption refers to the base power consumption of the node when it is idle mode, i.e., when no requests are being processed.

DPCR: Dynamic power consumption per request refers to the amount of power consumed by the node to process a single request.

NQ: number of queries.

We used the metric provided by the iFogSim simulator to analyze the energy use. Figure 7 depicts the simulation results schematically, with the x axis representing the variation in the number of queries and the y axis representing the energy used in Watt. As shown in the simulation results our RMSAPCC method can be considered as environmental approach among the other ones.

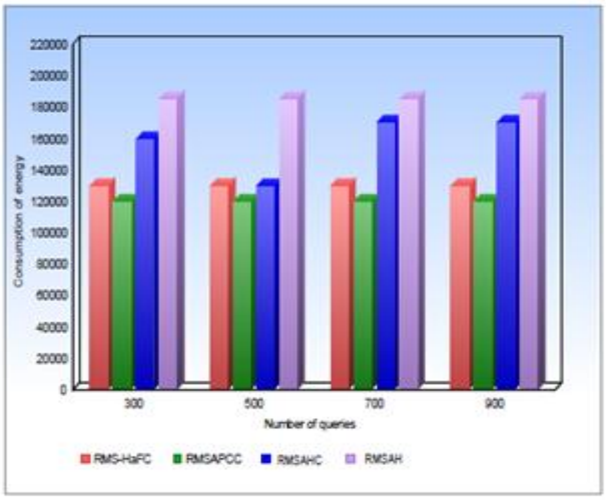


Figure 7: Consumption of energy

C.Network congestion

The formula used to calculate network congestion is as follows:

CN= number of service migrations * NQ (7)

In the series of simulations shown by Figure 9. The x-axis indicates the number of queries, and the y-axis depicts the amount of service transmitted through the network measured in KO, in all three techniques (RMSAH, RMSAHC, RMSAHCC) we see a difference in the use of the network. It is worth noting that the iFogSim simulator has this measure as well. The RMSAH and RMSAHC methods have lower utilization of the network than the RMSAPCC approach. We can justify this by that fog nodes exchange more information in a wide range. Despite this, our method RMSAPCC still better than RMS-HaFC method.

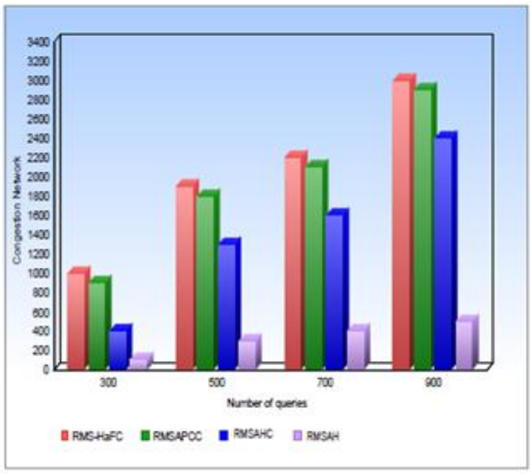


Figure 8: Congestion Network.

D. Additional metric S in a specific query number.

This additional metric is calculated by the sum of percentage of previous three metrics. Our RMSAHCC method offers an advantage by evaluating the three metrics with the same number of queries (700). This demonstrates the effectiveness of our RMSAHCC method.

We compared all strategies to the RMSAHCC strategy, Table 3 summarizes the obtained. It can be noticed that the RMSAHCC approach was able to improve the response time with a system equivalent to 30 ms compared to the RMS-HaFC, 30 ms compared to the RMSAHC strategy and 6.71 ms compared to the RMSAH strategy.

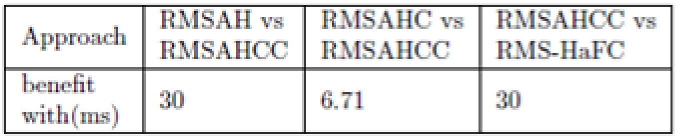
Table 4 shows the difference between three techniques (RMSAH, RMSAHC, RMS-HaFC) with RMSAHCC where we can deduce that our RMSAHCC approach reduced the energy with an average benefit of 10000 compared to the RMS-HaFC approach, an average benefit of 30000 compared to the RMSAHC approach and an average benefit of 50000 compared to the RMSAH approach. Therefore, our proposal RMSAHCC is much better compared to other approaches, it allows consuming less energy. It can be said that the energy consumption in all experiences is due to the increase in the number of queries.

Table 5 shows the difference between the three approaches (RMSAH, RMSAHC, RMS-HaFC) and RMSAHCC in terms of network load. We can conclude that RMSAH and RMSAHC approaches significantly reduced network usage with an average benefit of 600 and 1800 respectively. Despite this RMSAHCC gives good results compared to the RMS-HaFC approach, with an average benefit of 200.

We also estimated the benefit of our approaches compared to RMS-HaFC (see Table 6). From this table, we notice that our RMSAHCC approach gives a very significant benefit by considering the three metrics. It is therefore compared to RMS-HaFC, which offers a benefit of 11.7%.

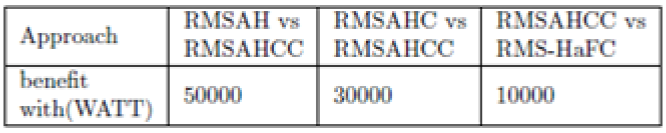
We also noticed in the simulation figures that the shape of the curve and the graphs indicate that the RMSAHCC method is positioned below the RMS-HaFC approach. This shows that it performs well, especially when considering the three metrics.

Table 3: Comparison of the response time.



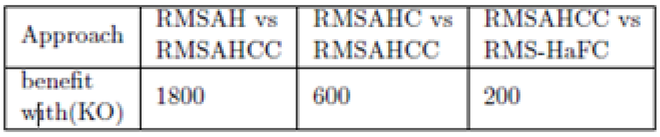
Approach	RMSAH vs RMSAHCC	RMSAHC vs RMSAHCC	RMSAHCC vs RMS-HaFC
benefit with(ms)	30	6.71	30

Table 4: Comparison between the energy consumption of the four approaches



Approach	RMSAH vs RMSAHCC	RMSAHC vs RMSAHCC	RMSAHCC vs RMS-HaFC
benefit with(WATT)	50000	30000	10000

Table 5: Comparison by related to network usage.



Approach	RMSAH vs RMSAHCC	RMSAHC vs RMSAHCC	RMSAHCC vs RMS-HaFC
benefit with(KO)	1800	600	200

Table 6: Summary of three metrics

Approach	RMSAH vs RMSAHCC	RMSAHC vs RMSAHCC	RMSAHCC vs RMS-HaFC
benefit (percentage)	0.71%	4.27%	11.7%

Vi.Conclusions:

Since the Cloud is a centralized paradigm located far from the users at the core of the network, the infrastructures of this computing model face problems of high latency and immense network traffic. Because of these difficulties encountered by the Cloud, the Fog computing paradigm has emerged. The fog infrastructures being vast, and geo-distributed, and since the localization of the services is crucial for the processing, we have studied how the services must be organized and managed in these infrastructures in order to reduce the latencies in the system, and reduce the network overload.

To this end, in this work, we have proposed a diagram which is used to route the query to the nearest node hosting the service requested by a user, possibly a connected object while balancing the workload. To reduce the average response time and optimize the total system energy consumption, we proposed a migration and replication strategy which we called it "RMSAPCC": Replication and Migration Based on Service accessibility Prediction with Connectivity based on Clusters. This strategy is an extension of our previous work. Indeed, it specifies the number and the best locations of replicas in each cluster. We evaluated our method with the IFOGSIM simulator, the results were positive.

References

[1] L. Atzori, A. Iera and G. Morabito, The internet of things: A survey, Computer Networks 54(15) (2010), 2787–2805.

[2] Y.S. Patel and T. Parmar, Cloud of things: A state-of-the-art review on integration of internet of things with cloud computing, IJCA Proceedings on National Conference on Contemporary Computing NCCC 2016(3) (2017), 37–41.

[3] B. Zhang, N. Mor, J. Kolb, D.S. Chan, K. Lutz, E. Allman, J. Wawrzynek, E.A. Lee and J. Kubiatowicz, The Cloud is Not Enough: Saving IoT from the Cloud, in: 7th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud '15, Santa Clara, CA, USA, July 6–7, 2015.

[4] M. Taneja and A. Davy, Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm, in: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, May 8–12, 2017, pp. 1222–1228.

[5] R. Mahmud, R. Kotagiri and R. Buyya, Fog Computing: A Taxonomy, Survey and Future Directions, in: Internet of Everything – Technology, Communications and Computing, B.D. Martino, K. Li, L.T. Yang and A. Esposito, eds, Springer, 2018, pp. 103–130.

[6] T. Huang, W. Lin, Y. Li, L. He and S. Peng, A latency-aware multiple data replicas placement strategy for fog computing, Journal of Signal Processing Systems 91(10) (2019), 1191–1204.

[7] A. Aral and T. Ovatman, A decentralized replica placement algorithm for edge computing, IEEE Transactions on Network and Service Management 15(2) (2018), 516–529.

[8] M.I. Naas, J. Boukhobza, P.R. Parvédy and L. Lemarchand, An Extension to iFogSim to Enable the Design of Data Placement Strategies, in: 2nd IEEE International Conference on Fog and Edge Computing, ICFEC 2018, Washington DC, USA, May 1–3, 2018, 2018, pp. 1–8.

[9] R. Vales, J. Moura and R.N. Marinheiro, Energy-aware and adaptive fog storage mechanism with data replication ruled by spatio-temporal content popularity, Journal of Network and Computer Applications 135 (2019), 84–96.

[10]. Berkennou, A., Belalem, G., Limam, S. (2020). A replication and migration strategy on the hierarchical architecture in the fog computing environment. Multiagent and Grid Systems, 16(3), 291-307.

[11] A.R. Benamer, H. Teyeb and N.B. Hadj-Alouane, Latency-Aware Placement Heuristic in Fog Computing En-vironment, in: On the Move to Meaningful Internet Systems. OTM 2018 Conferences – Confederated Interna-tional Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22–26, 2018, Proceedings, Part II, 2018, pp. 241–257.

[12] Khawla, B., Said, B., Mehdi, B. (2021, October). Replication and Migration Heuristics for Enhancing Service Accessibility in Fog Environment. In 2021 International Conference on Networking and Advanced Systems (ICNAS) (pp. 1-5). IEEE.