Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Highlights

Future Generation Computer Systems xxx (xxxx) xxx

CLARA: A novel clustering-based resource-allocation mechanism for exploiting low-availability complementarities of voluntarily contributed nodes

Sergio Gonzalo[*], Joan Manuel Marquès, Alberto García-Villoria, Javier Panadero, Laura Calvet

- Efficient resource allocation of volunteer resources is key for service placement.
- A clustering-based complementary availability mechanism of low-available nodes.
- Our mechanism harnesses availability complementarities for service placement.
- Our mechanism increases service capacity while reduces high-available nodes overload.
- Lazy node reassignment algorithm minimizes reassignments in complementary nodes.

FUTURE: 6280

ARTICLE IN PRESS

Future Generation Computer Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

CLARA: A novel clustering-based resource-allocation mechanism for exploiting low-availability complementarities of voluntarily contributed nodes

Sergio Gonzalo [a,*], Joan Manuel Marquès [a], Alberto García-Villoria [b], Javier Panadero [a], Laura Calvet [a]

[a] IN3 - Computer Science Dept., Open University of Catalonia, Barcelona, Spain
[b] Dpto. de Organización de Empresas, Instituto de Organización y Control, Universitat Politècnica de Catalunya, Spain

A R T I C L E I N F O

Article history:
Received 11 May 2021
Received in revised form 2 October 2021
Accepted 4 October 2021
Available online xxxx

Keywords:
Resource allocation methods
Volunteer-Computing
K-means clustering
Low-availability complementary relationships

A B S T R A C T

Volunteer Computing is a type of large-scale distributed system formed aggregating computers voluntarily donated by volunteers. These computers are usually off-the-self heterogeneous resources belonging to different administrative authorities (users) that have an uncertain behavior regarding connectivity and failure. Thus, the resource allocation methods in such systems are highly dependent on the availability of resources. On one hand, resources tend to be scarce, but on the other hand, computers exhibiting low availability patterns – which are the most frequent type – are discarded or used at a high cost only when high available nodes are crowded. This paper presents the Complementary Low-Availability Resource-Allocation (CLARA) mechanism, a novel clustering-based resource allocation mechanism that takes advantage of complementarities between nodes with low availability patterns. The combination of them into complementary nodes offers an availability level equivalent to the level offered by a single high-available node. These groups of complementary nodes are maintained using a lazy reassignment algorithm. Consequently, a significant number of nodes with low-availability patterns are considered by the resource allocation mechanism for service placement. Our method has been validated over a simulation environment of a real volunteer network. The analysis of the results shows how our mechanism maximizes the use of poor quality computational resources to satisfy the user quality requirements while minimizes the number of USs replicas reassignments between nodes. As well, the capacity of the system for providing user services is highly increased while the load of the high-available nodes is remarkably reduced.

1. Introduction

A volunteer network is a type of large-scale distributed network where the computational resources are voluntarily contributed by the participants in the network. These resources are normally based on spare computing resources which are often inexpensive off-the-shelf equipment with a high degree of heterogeneity. In consequence, these resources have an uncertain behavior regarding connectivity and failure. The inherent challenges of these types of systems (data loss, data integrity, resources unavailability, and replication costs) traditionally lead to prioritize the usage of the most available nodes. In this context, the Resource Allocation (RA) mechanisms are highly dependent on the availability of the underlying resources. The scarcity of reliable resources jointly with the high number of nodes exhibiting low availability patterns (which is the most frequent type) force to choose between discarding the low-available ones (leading to a crowding of the most available nodes) or alternatively, using them at a high cost (replication to guarantee reasonable availability levels is costly).

In this context, our motivation is to address the problem of developing a RA mechanism that allows considering most of these low-available nodes (traditionally discarded or used at high cost) as part of the pool of resources for service deployment, increasing the overall capacity of the network. The main goal is to ensure the Quality of Service (QoS) required by a user service (US) deployed on low-available nodes. The complementarities between the availabilities of the nodes with low availability patterns allow offering additional QoS levels for USs deployed on low-available nodes. The secondary goal is to minimize the number of USs replicas reassignments (replications) on low-available nodes. Notice that the incorporation of low-available nodes into the RA

* Corresponding author.
E-mail addresses: sgonzalos@uoc.edu (S. Gonzalo), jmarquesp@uoc.edu (J.M. Marquès), alberto.garcia-villoria@upc.edu (A. García-Villoria), jpanaderom@uoc.edu (J. Panadero), lcalvetl@uoc.edu (L. Calvet).

https://doi.org/10.1016/j.future.2021.10.002
0167-739X/© 2021 Elsevier B.V. All rights reserved.

pool will increase the number of unavailabilities of these nodes, and therefore, the unavailability of the USs deployed on them. Finally, the third goal is to maximize the usage of the volunteer resources, increasing the overall capacity of the network for USs placement.

In this paper, by combining availability prediction and clustering techniques, we propose a RA methodology approach, and then consequently, develop a new RA mechanism over low-available voluntarily contributed resources. The main contributions of this paper can be summarized as follows:

- A novel clustering-based RA mechanism that leverages complementarities between low-available nodes for US placement. These complementarities satisfy the same US requirements applied to high-available nodes.
- An algorithm for the creation of complementary sets of nodes. The algorithm makes use of clustering techniques for grouping low-available nodes into disjoint sets with similar availability patterns. Next, it creates combinations of nodes with complementary availability from these sets. The resulting combined node, named complementary node (CN), provides an availability level similar to the level offered by a high available node.
- A lazy reassignment algorithm that limits the reassignments to the minimum necessary to restore the missing US QoS, reducing the replication needs remarkably.
- An evaluation and comparison with a method based on the usage of the most available nodes only. This evaluation has been done using a simulation environment of a real-microblogging application named Garlanet.[1]

The remainder of this paper is structured as follows. Section 2 presents a literature review about RA methods used in distributed networks based on volunteer networks. The formal problem description is detailed in Section 3. The CLARA mechanism description and the methodology steps are described in Section 4. Section 5 describes the main algorithms developed as part of the CLARA mechanism. The simulation environment and the complete set of experiments and analysis are described in Sections 6 and 7, respectively. Finally, Section 8 draws the most relevant conclusions and identifies future research lines.

2. Related work

Volunteer Computing (VC) is a type of distributed computing in which non-dedicated computing resources are donated, providing an unrivaled computing capacity. However, this non-dedicated nature of the resources makes them extremely stochastic and unpredictable, and complex mechanisms based on costly redundancies must be deployed in order to guarantee reasonable availability levels [1,2]. The feasibility of running applications in contributory networks is studied by [3] which presents the analysis of several open-source applications deployed on two VC community networks. A micro-blogging application is one of this suitable applications, where users make use of shared resources to store and interchange data (usually in the form of snippets of a small number of characters) establishing virtual communication channels among them [4,5].

2.1. Resource allocation and service placement

The volatility of the nodes in a VC network leads to design smart RA strategies for Service Placement (SP) under strict restrictions of network efficiency, resources availability and cost, and service reliability and performance. Thus, it is necessary to deploy redundancy mechanisms and efficient policies according to specific criteria based mainly on node and application parameters as well as on the network status.

One RA research area is focused on *metaheuristics* for optimizing the usage of the contributed resources. Metaheuristics and simulation techniques have been widely used to address the stochastic nature of the RA problem of contributory resources in VC networks. [6] presents a methodology for extending metaheuristics through simulation to solve stochastic combinatorial optimization problems while facilitates the introduction of risk and reliability criteria during the assessment of alternative high-quality solutions. [7] proposes a heuristic method based on a weight system to determine resources qualities. Afterward, a biased random procedure allows selecting them accordingly in an extremely fast way. [8] presents a metaheuristic approach designed to deal with applications where data availability must be always guaranteed, including a simheuristic to handle the stochasticity of the resources. [9] studies the RA problem on Fog/Edge computing, proposing a heuristic-based RA algorithm inspired by an economic model. This model aims to maximize the number of applications served by the network while ensuring a target operational cost. Finally, other authors like [10] studies the problem from its analogy with other well-known heuristic problems, modeling the allocation problem as a bounded 0–1 multidimensional knapsack problem.

Other RA research area is focused on the usage of *reputation and trust models* since the performance of tasks over VC resources may be affected on their completion time due to the unreliability of the VC resources. [11] proposes a reputation and resource-based reliability model which uses a machine learning model to extract resource usage patterns from historical data to predict the reliability of the hosts. Similarly, in [12] the authors address the challenge to guarantee a minimum QoS based on the tracking and monitoring of the reliability and trust of the donated resources. The proposed model extends classical reputation models incorporating a probabilistic model which considers several fine-grained parameters for the reliability estimation of the untrusted resources. Finally, [13] proposes a trust-based scheduling approach where the trust level of each node is derived from its underlying performance while tasks are prioritized according to their resource requirements and cost. The tasks are smartly mapped to the correspondent node, minimizing the costs of processing the task.

Finally, other RA research areas are based on the impact on the *performance, integrity, and serviceability*. [14] presents a model to evaluate the system performance as a function of the nodes resources, the stochastic demand, and the servers vulnerabilities. [15] presents a method to address the underused resources and the additional costs on fixed RA methods. [16] defines a workflow for enhancing the usage of hybrid VC and cloud resources. [17] proposes a blockchain-based architecture to ensure the integrity of the resource transaction data. [18] introduces a blockchain-enabled resource sharing and service composition solution through volunteer computing. Device capabilities are made available for sharing using blockchain while miners are used for searching non-advertised service capabilities to ensure a fast and reliable service provisioning framework. [19] presents a RA system that considers the minimum availability level required by the user and the minimum cost to allocate resource. [20] proposes a self-adaptive RA method based on an iterative QoS prediction model and a runtime decision algorithm which improves the QoS value on each iteration. [21] combines the Fog Computing (FC) and VC paradigms, leveraging underutilized resources of end devices to address the high latency, energy consumption, and network usage for delay-sensitive applications. [22,23] develops replica decision policies for a VC micro-blogging service

[1] https://dpcs.uoc.edu/projects/garlanet/.

with a focus on maximizing the content availability and the number of replicas generated. Finally, emerging researches are related to vehicular VC. The recent advancements in vehicular communication technologies and the embedment of computing resources in vehicles lead to dispose a large number of powerful computing resources that can be employed for services deployment. [24] leverages the surplus of vehicles resources for proposing a hybrid VC-based model that allows to minimize the latency, maximize the system utility, and reduce the monetary costs for task offloading. [25] examines the characteristics presented by vehicular VC and the research challenges to be overcome for intensive computing projects. [26] presents a vehicles computing framework for offloading computing tasks for a better computation services. The authors analyze the interactions between vehicles and nodes designing a genetic algorithm to find the optimal node strategy. [27] presents a workload clustering-based resource provisioning mechanism for executing cloud-based applications with heterogeneous workloads. This mechanism combines biogeography-based optimization techniques with K-means clustering to classify the cloud workloads according to their QoS requirements. Finally, [28] presents a hybrid solution to handle resource provisioning using workload analysis in a cloud environment. The solution uses the imperialist competition algorithm and K-means for clustering the workload submitted by end-users and a decision tree algorithm for determining scaling decisions.

RA in VC networks is closely related to the *services placement* objective. Thus, [29] proposes a multicriteria optimization strategy for sorting and selecting the most suitable nodes. [30] presents a low-complexity bandwidth-aware SP heuristic, which leverages network state information to maximize the bandwidth allocation on micro-cloud deployments. [31] proposes a network-aware model based on a low-complexity SP heuristic that considers the limited capacity of the nodes, and the unpredictable network performance, for maximizing the bandwidth allocation. [32] proposes an availability and reliability prediction model based on a multi-state semi-Markov process for determining the most suitable VC nodes for SP. [33] presents a machine learning model to reduce the complexity of the SP. The model is complemented with and a set of readjustment processes, and an optimized k-medoids clustering approach.

As a summary, Table 1 presents the classification and characteristics of the related work.

Our research aims to provide a different RA approach through a clustering-based mechanism for the definition of complementary relationships of low-available nodes. Thus, our approach allows leveraging low-available nodes that individually would be discarded. As result, the whole list of eligible nodes for US placement is remarkably increased while the overload of the most available and demanding nodes is reduced.

3. Problem description

By community-owned VC systems, we refer to systems that host their data and services in computers voluntarily contributed by participants in the system. They are large-scale networks where computational resources spread across the Internet. Thus, no central authority is responsible for providing the resources which are often inexpensive off-the-shelf equipment with a high degree of heterogeneity. In this context, one suitable US is a decentralized micro-blogging application where the user data is kept in microservices. These microservices are distributed and replicated among VC nodes (see Fig. 1).

However, resource sharing is voluntary which entails a lack of reliability in nature. Thus, the most available nodes are prioritized, discarding a non-negligible number of nodes exhibiting low availability patterns. Besides, each US must be replicated

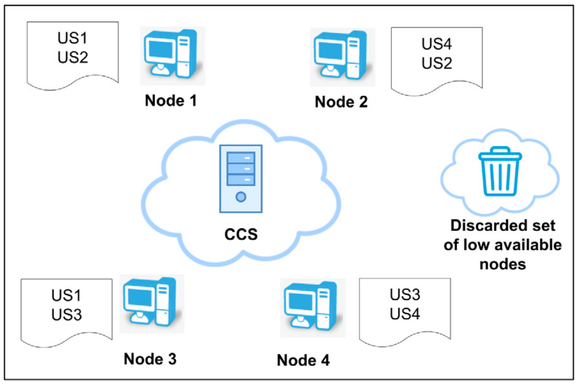


Fig. 1. General US deployment scenario over volunteer resources.

in several nodes to provide ways to access the US information in case of the unavailability of one node. This creates a cost in storage, time, bandwidth, and latency which must be minimized. In summary, the balance between cost and availability will be one of the most important objectives of the problem.

To orchestrate all the deployment and replication processes, this type of applications incorporates a Centralized Control System (CCS) component that decides which US should be managed by each node and also informs which nodes manage each US when it is asked about. In addition, the CCS knows which nodes are available and decides which US must be replicated on which nodes so that all US information is always available.

The challenge is to design a RA mechanism that maximizes the usage of the donated resources. Concretely, the mechanism should leverage donated resources that exhibit a low-availability profile for increasing the eligible set of resources for US deployment, according to the following premises:

- **Maximize the US quality**: We consider that the US QoS is proportional to its availability. Thus, node availability is the most important QoS parameter, so nodes that are most of the time available should be prioritized.
- **Minimize the number of US replicas**: When a new US is created (at US deployment time), US data is replicated over several nodes in the network. The number of US replicas will depend on the availability of the underlying nodes. As nodes are more available, a fewer number of replicas are needed. Thus, the goal is to allocate reliable resources for minimizing the number of replicas to use.
- **Minimize the number of US replicas reassignments**: If one of the US replicas gets unavailable, a new replica must be enabled in a new available node. This new replica is built from one of the surviving replicas, copying the US data to the new node. However, this operation is costly in terms of (1) US data size (which depends on the US characteristics), (2) bandwidth and latency between origin and destination nodes, (3) node storage (same storage is consuming on several nodes, enough storage is needed on destination node), and (4) timing (US may be running in a degraded mode because not all the required replicas are available until the copy is finished). Therefore, the goal is to minimize the number of reassignments for restoring the missing US QoS in case of nodes unavailability.
- **Maximize the use of low-available resources**: The mechanism must avoid overloading the most available nodes while maximizes the use of low-available nodes.
- **Maximize the distribution of resources among all users**: Computational resources must be smartly distributed among users to maximize its usage.

Table 1
Related work review summary.

Techniques	Evaluation tools	Workload type	Performance metrics	Advantages/Disadvantages
- Metaheuristics - Heuristics - Simheuristics	- Owned simulators - Sim-Opt integration - Monte Carlo simulation - Discrete event simulation	- Analytical expressions - Problem-specific data	- Risk and reliability metrics - Alternative solutions of similar HQ - Weights quality tuning - System stochasticity quantification	Advantages: - Short computing time for HQ solutions Disadvantages: - Deterministic assumptions - Performance depends on adequate parameter tuning
- Reputation and trust models	- Machine learning-based simulators - Bayesian method-based reputation - Probabilistic models for reliable estimations - Owned simulators	- Host and tasks historical data - Resources usage trace data - Training data	- Usage patterns - Task failure rates - Reliability predictions	Advantages: - Modeling of resources behavior - Resources prioritization by quality-sorting Disadvantages: - Need of track resources reliability - Trust monitoring of distributed resources - Coarse-grained reliability models
- Performance, integrity, and serviceability models	- Facility location theory - Fuzzy models-based reliability prediction - Redundancy models - Queueing theory-based arrival models	- Hosts and tasks historical data - Task types distributions	- Completion time - Rework rates - Cost metrics - Expected reliable resources - Variability and resources correlation	Advantages: - Service-oriented tasks maximization - Data-driven scheduling Disadvantages: - Non-convex stochastic problems requires both lower and upper bounds for reducing state-space

Each of the US to place on the allocated resources is defined by a required US QoS (Q_{US}) which is distributed among the high-available and the low-available sets of nodes according to a distribution parameter (α) such as:

- α% of the required Q_{US} is provided by high-available nodes (Q_{USHQ}).
- (1-α)% of the required Q_{US} is provided by combined groups of low-available nodes (Q_{USCN}).

Thus, the required Q_{US} is the sum of the qualities provided by high-available nodes and low-available combined nodes.

$$Q_{US} = Q_{USHQ} + Q_{USCN}$$

Accordingly, the value of the α parameter is one of the optimization parameters to be included in the model.

3.1. Problem definition

Under these premises, the objective is to find a near-optimal way to group low-available nodes into complementary relationships to provide a combined availability equivalent to the one exhibited by a high-available node. These new relationships will enable these resources for SP, minimizing the replicas reassignments, reducing the occupancy of the high available nodes, and maximizing the usage of all the nodes in the network. Thus, we can formulate the problem as follows:

Find out the minor set of nodes, $CN \subseteq C$, that:

$$CN = \underset{CN' \subseteq C}{\arg\min} \mid CN' \mid \tag{1}$$

subject to:

$$Q_{CN} = \frac{\sum_{h\in H}\sum_{c\in CN}(\min(|W|\cdot 60, K_{c,h}))}{|W|\cdot|H|\cdot 60} \geq T \tag{2}$$

$$\sum_{c\in CN}(\min(|W|\cdot 60, K_{c,h})) \geq S\cdot|W|\cdot 60, \forall h \in H \tag{3}$$

Table 2
Notations used in problem description.

Symbol	Significance
Indices	
H	List of timeframes (hours) in a week: $H = \{1, \ldots, 168\}$.
$\|H\|$	Number of timeframes (hours) in a week.
Parameters	
C	Set of candidate nodes.
T	Quality threshold for a high-available node ($T \in \{0, 1\}$).
S	Timeframe minimum availability threshold ($S \in \{0, 1\}$).
L	Minimum quality threshold required for a node ($L \in \{0, 1\}$).
W	Historical nodes availability information grouped by week.
$\|W\|$	Number of weeks used for the availability prediction.
Q_{US}	Required quality for a user service (US).
Q_{USHQ}	Chunk of Q_{US} on high-available nodes.
Q_{USCN}	Chunk of Q_{US} on combined low-available nodes (CN).
Q_c	Quality of a single node.
Q_{CN}	Quality of a complementary node.
α	Percentage of distribution of the Q_{US} into Q_{USHQ} and Q_{USCN}.
$K_{c,h,w}$	Available minutes exhibited by node c in the timeframe h of the week w ($c \in C, h \in H, w \in W$).
$K_{c,h}$	Available minutes exhibited by node c in the timeframe h during the AP interval. $\sum_{w\in W} K_{c,h,w} (c \in C, h \in H, w \in W)$.
Variables	
CN	Complementary nodes relationships

Formula (2) enforces the CN to exhibit a quality equivalent to the quality of a single high-available node, i.e., its combined quality Q_{CN} must be equal to or greater than the threshold T. Formula (3) enforces the CN to exhibit a minimum availability prediction on at least one timeframe of the week, i.e., the combined AP built from all the nodes participating in the complementary node must be equal to or greater than the threshold S for at least one timeframe (see Table 2).

4. Low-available nodes clustering-based RA mechanism

The CLARA mechanism presented in this paper aims to leverage VC nodes exhibiting a low-availability profile for increasing the eligible set of resources for SP. The mechanism relies on

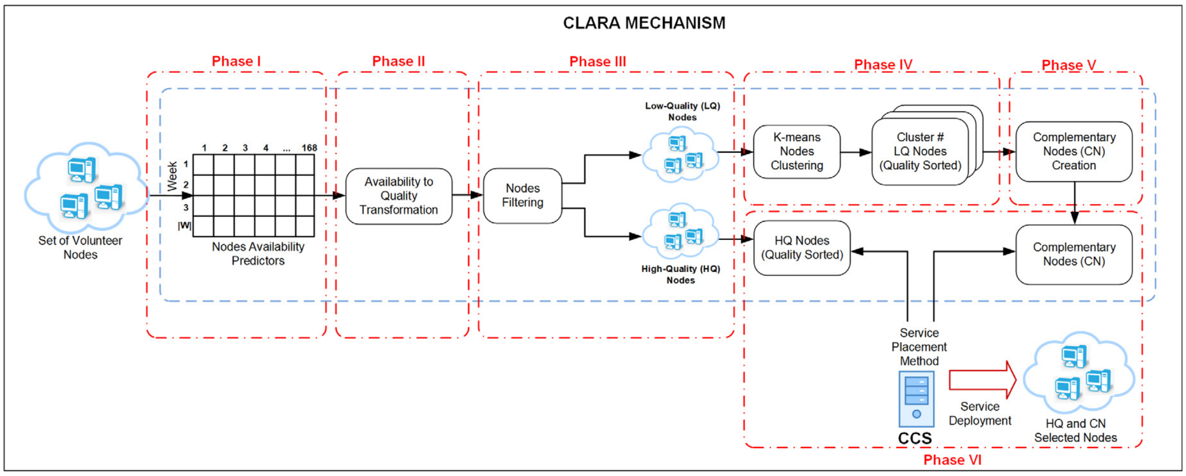


Fig. 2. Representation of the CLARA mechanism methodology phases.

clustering techniques for creating disjoint sets of low-available nodes with complementary availability. As result, the capacity of the network for SP is increased while the overloading of the high-available nodes is highly decreased.

Fig. 2 presents the CLARA methodology phases. The methodology is composed of 6 phases which can be gathered into two different groups. On one hand, phases I to IV are executed periodically for updating the clusters and sorting the list of nodes by their quality. This group aims to keep clusters up-to-date based on nodes behavior over time in order to be later efficient for finding complementary relationships between nodes. On the other hand, phases V and VI work with results from the previous phases to allocate resources for new US placements when required. Fig. 3 shows the phase sequence of the mechanism and the interactions between the CLARA components.

4.1. Phase I: Definition of the nodes availability predictor

Following the study performed in [34] where the problem of predicting if a group of resources may be continuously available for a relatively long time period, our mechanism defines an Availability Predictor (AP) to characterize the availability of the nodes in the VC network. [34] showed how their prediction methods can reliably guarantee the availability of collections of VC resources, and how this prediction is particularly useful for SP on VC networks. Thus, the node availability can be summarized as a vector of 168 positions (24 h per day, 7 days in a week) where each position represents the probability that the node is connected at that specific hour in the following weeks.

According to the results also exhibited on [34], the prediction interval is based on the availability exhibited by the nodes during the last 4 weeks *i.e*. the number of previous weeks to be used for predicting the node availability in the following weeks. Therefore, each node will be identified by an availability vector of 168 positions on which each position will range from 0 to 240 min (60 min/h x 4 weeks).

Thus, for each node, the CCS records the total number of minutes that the node has been available at each week timeframe (hour) on each of the previous weeks used for prediction (AP interval). As result, each node will be characterized by an AP vector of 168 positions where each position stores the total number of minutes that the node has been available at that specific timeframe during the prediction interval.

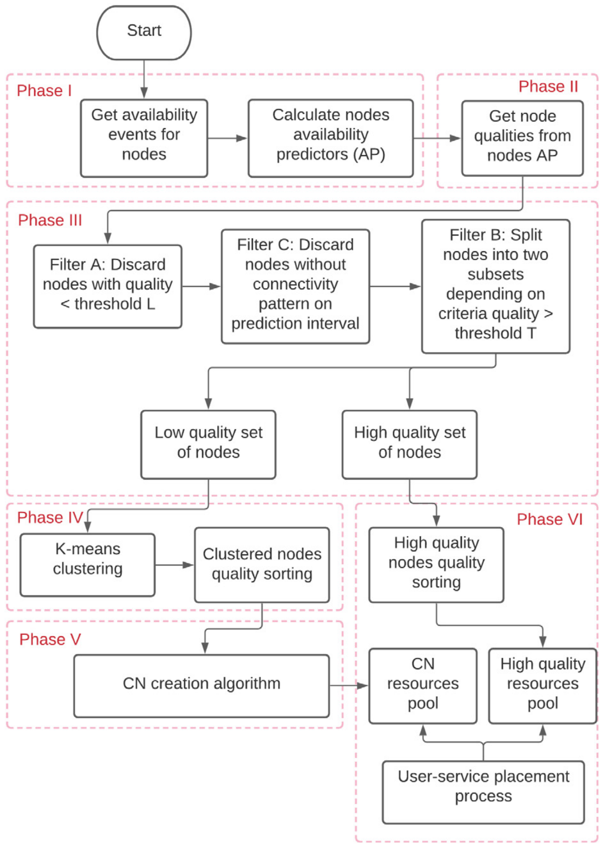

Start
Phase I
Get availability events for nodes
Calculate nodes availability predictors (AP)
Phase II
Get node qualities from nodes AP
Phase III
Filter A: Discard nodes with quality < threshold L
Filter C: Discard nodes without connectivity pattern on prediction interval
Filter B: Split nodes into two subsets depending on criteria quality > threshold T
Low quality set of nodes
High quality set of nodes
Phase IV
K-means clustering
Clustered nodes quality sorting
Phase VI
High quality nodes quality sorting
Phase V
CN creation algorithm
CN resources pool
High quality resources pool
User-service placement process

Fig. 3. CLARA mechanism phase sequence and components interaction.

4.2. Phase II: Transformation of nodes availability prediction to quality

The node AP is converted into a single numerical value which represents the quality of the node. This transformation aims to simplify the nodes sorting process, speeding up the processing on the subsequent phases.

The translation from the node AP to a quality value aims to obtain a normalized value, in the range [0,1], based on the exhibited availability of the node during the prediction interval. The quality is calculated as the average of the percentages of time that the node has been available on each week timeframe during the AP interval, according to the following expression:

$$Q_c = \frac{\sum_{h \in H} K_{c,h}}{|W| \cdot |H| \cdot 60} \tag{4}$$

4.3. Phase III: Filtering and separation of nodes into high-quality and low-quality nodes

The filtering process aims to remove nodes with a random or unpredictable behavior exhibited during the AP interval. As well, nodes exhibiting a very low connection profile are also removed with the aim to simplify the identification of availability patterns on the whole set of filtered nodes.

The filtering process considers three types of sub-filters:

- **Filter A**: Discard nodes with a quality below a minimum required threshold (L). This aims to discard nodes exhibiting a very low connection profile during the week.
- **Filter B**: Discard nodes with quality over the high-available threshold (T). This filter separates nodes exhibiting a very good connection profile (high-quality nodes) from the rest of the nodes (low-quality nodes).
- **Filter C**: Discard nodes that do not exhibit a connectivity pattern along all the weeks in the AP interval. This filter discards nodes with a very changing availability pattern during the evaluation period and, therefore, with an "unpredictable availability pattern" (threshold S).

As result, the filtered nodes are separated into two sets:

- High Quality (HQ) nodes, which are characterized to have a quality equal or greater to a specific threshold (T).
- Low Quality (LQ) nodes, which are characterized to have a quality between the minimum quality threshold (L), and the high-available threshold (T). As well, they exhibit a connectivity pattern during their AP interval, i.e. a quality greater than threshold S on at least one timeframe of the AP. These nodes are the ones considered for clustering, and later, finding complementary availabilities.

4.4. Phase IV: Clustering of low-quality nodes according to their availability prediction

The objective of this phase is to group LQ nodes into disjoint sets of nodes for later simplifying the process of combining them complementarily (see Figs. 2 and 3). Our mechanism considers the LQ nodes APs as the set of observations to cluster. The AP is a two-dimensional vector of 168 positions that stores the number of minutes that the node has been available at each timeframe during the AP interval.

The output of the clustering process will be sets of LQ nodes grouped into clusters. The APs of the nodes in the same cluster will be very similar between them, but quite different from the rest of APs on other clusters. In consequence, the processing time of the algorithm for finding complementary availabilities in the subsequent phases is considerably reduced.

The K-means clustering algorithm has been used, evaluating a wide-enough range of K values to select the most appropriated one. The selection of the best K value, among all the evaluated ones, is validated by the usage of different clustering indices and metrics such as the Dunn, Calinski–Harabasz, and Davies–Bouldin indices.

4.5. Phase V: Identification and creation of set of nodes with complementary availability (complementary nodes)

A Complementary Node (CN) will be formed by LQ nodes which, combined between them, may act as good candidates for SP. At this regard, the clustering of nodes is fundamental. The clustering process allows to group LQ nodes according to their AP, laying the foundations for building CNs based on the selection of LQ nodes from the resulting clusters. The combination of several LQ nodes for building a CN firstly requires declaring a combined quality (CN quality) based on the following steps:

a. Each LQ node added to the CN contributes with its AP into a combined AP.
b. The CN quality is obtained from the "availability-to-quality" transformation performed on the combined AP.
c. If the resulting CN quality is lower than the minimum required quality for a HQ node (threshold T), then continue adding nodes to the set. Otherwise, the set is closed. The CN is built with the list of LQ nodes of the set and the resulting CN quality.

The CN quality evolves over time depending on the current state of the underlying LQ nodes in the CN:

- If all the LQ nodes in the CN are disconnected, independently of whether they should be connected or not at the current timeframe, the CN quality is computed as 0.
- If there is at least one LQ node connected in the CN at the current timeframe, the CN quality will depend on the status of the LQ nodes:
  - Connected LQ nodes and disconnected LQ nodes (nodes that should be disconnected at the current timeframe according to their AP) are considered for the CN quality calculation.
  - Failed LQ nodes (disconnected nodes that should be connected at the current timeframe according to its AP) are skipped for the CN quality calculation.

Finally, the selection of nodes from each cluster is performed using a descending sorted list by the node quality. The CN selection algorithm will select LQ nodes from the sorted list of each cluster using biased randomization techniques.

4.6. Phase VI: Selection of nodes for SP on high-quality and complementary nodes

Once the CNs have been created, the selection algorithm will proceed to select HQ and CN nodes for SP. Each of the USs is characterized by a required QoS (Q_{US}) which is a combination of the individual qualities offered by each type of node:

$$Q_{US} = Q_{USHQ} + Q_{USCN}$$

To later study the sensitivity in the distribution of user quality among HQ and CN nodes, the mechanism considers a distribution parameter α % such that:

$$Q_{US} = \alpha\% * Q_{USHQ} + (1 - \alpha\%) * Q_{USCN}$$

The selection algorithm will start selecting HQ nodes until reaching the minimum HQ required quality Q_{USHQ}. Then, the selection algorithm will continue selecting CNs until reaching the minimum CN required quality Q_{USCN} (see Fig. 4).

5. Algorithms

This section describes the main algorithms developed as part of the CLARA mechanism.

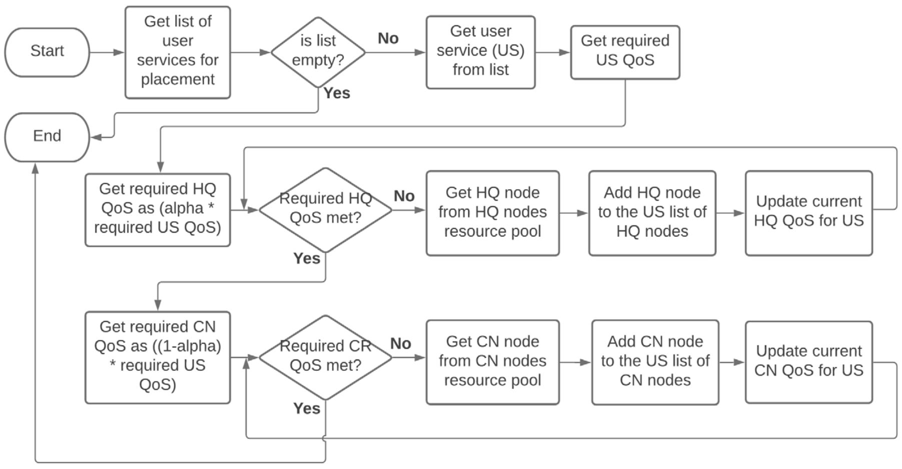

Start
Get list of user services for placement
is list empty?
No
Yes
Get user service (US) from list
Get required US QoS
End
Get required HQ QoS as (alpha * required US QoS)
Required HQ QoS met?
No
Yes
Get HQ node from HQ nodes resource pool
Add HQ node to the US list of HQ nodes
Update current HQ QoS for US
Get required CN QoS as ((1-alpha) * required US QoS)
Required CR QoS met?
No
Yes
Get CN node from CN nodes resource pool
Add CN node to the US list of CN nodes
Update current CN QoS for US

Fig. 4. Nodes selection process for user services placement.

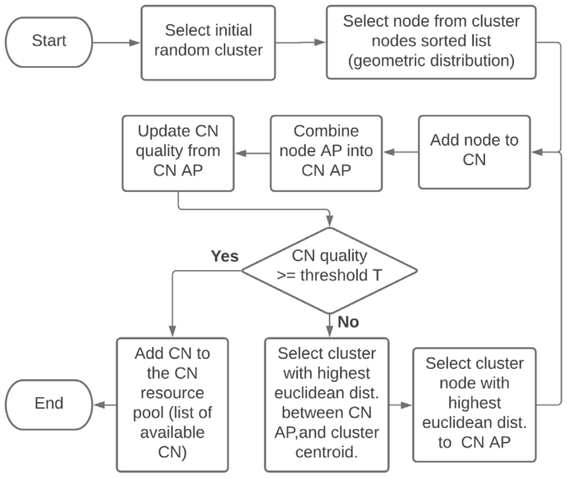

Start
Select initial random cluster
Select node from cluster nodes sorted list (geometric distribution)
Update CN quality from CN AP
Combine node AP into CN AP
Add node to CN
Yes
CN quality >= threshold T
No
End
Add CN to the CN resource pool (list of available CN)
Select cluster with highest euclidean dist. between CN AP,and cluster centroid.
Select cluster node with highest euclidean dist. to CN AP

Fig. 5. Complementary nodes creation algorithm flowchart.

5.1. Complementary nodes creation algorithm

The CN creation algorithm (see algorithm 1) is responsible for building up a CN node from LQ nodes selected from the different clusters. This CN node will be characterized by the list of LQ nodes and a combined quality (CN quality).

The algorithm (see Fig. 5) starts selecting one active LQ node from the sorted list of LQ nodes of a randomly selected initial cluster. The LQ nodes sorted list for each cluster is a descending list based on the node quality. This list is updated periodically depending on the connection status of the nodes. The selection of nodes from the sorted list is performed using a geometric distribution that will prioritize nodes with better quality, avoiding the overloading of LQ nodes with the highest quality. Once the LQ node from the first cluster has been selected, the algorithm selects the cluster with the highest Euclidean distance between the combined AP (sum of the APs from all the nodes participating in the CN) and the rest of cluster centroids. The selection of LQ nodes in the selected cluster is again based on the highest Euclidean distance between the combined AP and the APs of the cluster LQ nodes to locate the LQ node with the best complementarity. On each iteration, the combined AP and quality are updated. The process is repeated until reaching a combined quality equal to or greater than the quality required for a single HQ node (threshold T).

5.2. User service replicas reassignment algorithm

Volunteer nodes are connected and disconnected over time according to a pattern characterized by their correspondent AP. In case a node gets disconnected, the hosted USs cannot be served and the US replicas must be reassigned to another connected node to ensure that the required USs QoS is satisfied.

However, the US replicas reassignment process depends on several factors such as the type of failed node (HQ or CN), the unavailability cause, and the status of the remaining LQ nodes (in the case of a CN node). At this regard, we distinguish between the concept of “failed" and “not connected" nodes. A “not connected“ node represents a node that is disconnected at the current timeframe, but this is its expected status in that timeframe according to its AP. In this case, the node QoS is considered for computing the CN quality. On the other hand, a “failed” node is a node that is disconnected at the current timeframe, but this is not its expected status according to its AP. In this case, the node QoS is not considered for computing the CN quality. Additionally, the concept of “expectation to be connected” is implemented as the total number of minutes exhibited by a node in the specific timeframe during the AP interval. If this total number of minutes is equal to or higher than the threshold S, the node is “expected to be connected”. Otherwise, the node is “expected to be disconnected”.

Fig. 6 provides a high-level specification of the US replicas reassignment algorithm. While the required US QoS is not met, the algorithm starts checking the HQ nodes assigned to the US. If any of the HQ nodes is disconnected, it is replaced, and a US replica is assigned to the new HQ node. Next, the algorithm checks the CNs assigned to the US. For each CN, it firstly computes the current CN quality. If this CN quality is not equal to or greater than the threshold T, the CN restoring process is started. If all the LQ nodes in the CN node are disconnected, the quality is computed to 0, and a complete CN reassignment is performed (a US replica is assigned to all the LQ nodes of the new CN). On the other hand, if there is, at least, one connected node in the CN, failed nodes are replaced one by one. Then, US replicas are assigned to the new LQ nodes until the QoS of the CN reaches the threshold T. Thus, the US replicas reassignment algorithm implements a **lazy replacement method** to ensure that only the minimum number of LQ nodes are replaced for restoring the missing US QoS.

Algorithm 2 contains a detailed description of the algorithm. To simplify the algorithm specification, the error and validation tasks have been excluded from the specification.

Algorithm 1: Complementary Nodes Creation Algorithm

```
Input: The minimum required quality for considering a single CN node as a HQ equivalent node: ThresholdHQ ; The minimum total quality
       required to be covered by CN nodes: CNsRequiredQuality
Output: The list of CN nodes: CNsList; The total CN quality for the corresponding list of CN nodes: CNsQuality
Procedure ComplementaryNodesCreation:
    CNQuality ← 0
    while CNsQuality < CNsRequiredQuality do
        Quality ← 0
        NodesList, AggregatedWeekPredictor, CNList ← empty list
        clIndex ← get random cluster number
        FirstNode ← selectCNnode(clIndex)
        FirstNodeWeekPredictor ← get AP from FirstNode
        NodesList ← add FirstNode
        AggregatedWeekPredictor ← add FirstNodeWeekPredictor
        Quality ← calculate Quality for AggregatedWeekPredictor
        while (Quality < ThresholdHQ) do
            SelectedNode, SelectedNodeWeekPredictor ← empty
            clIndex ← select cluster with highest Euclidean distance (clCentroids, AggregatedWeekPredictor)
            ListActiveNodes ← get the active nodes sorted list from cluster clIndex
            SelectedNode ← select node with highest Euclidean distance (ListActiveNodes, AggregatedWeekPredictor)
            SelectedNodeWeekPredictor ← get AP from SelectedNode
            NodesList ← add SelectedNode
            AggregatedWeekPredictor ← add SelectedNodeWeekPredictor
            Quality ← update Quality for AggregatedWeekPredictor
        end
        CN ← create CN from NodeList and Quality
        CNsList ← add CN
        CNsQuality ← increment by Quality
    end
return CNsList, CNsQuality
```

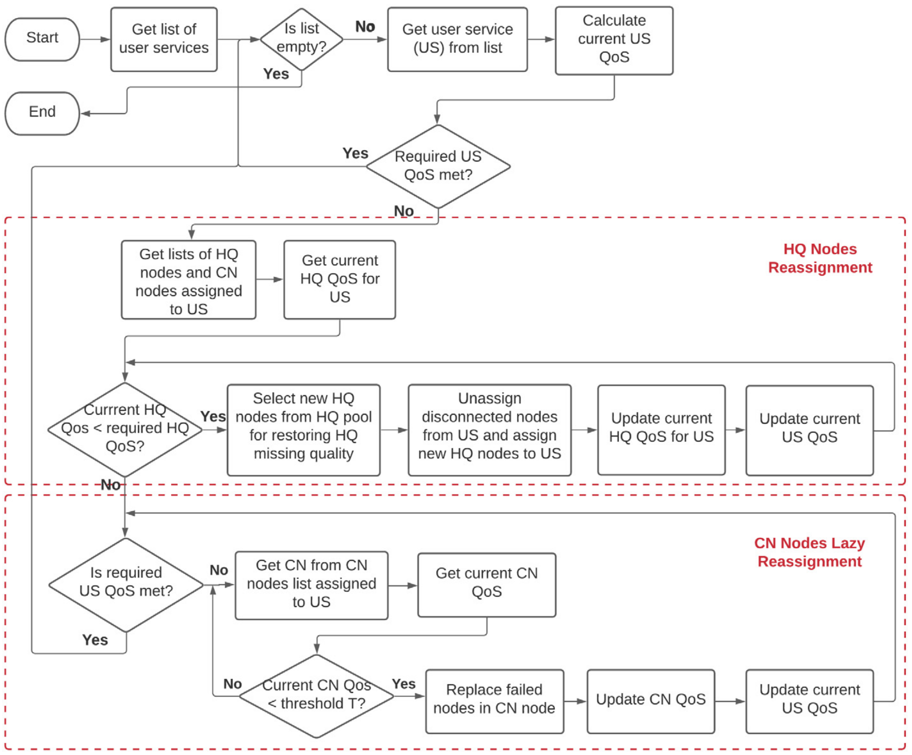

Start
Get list of user services
Is list empty?
No
Yes
Get user service (US) from list
Calculate current US QoS
End
Yes
Required US QoS met?
No
HQ Nodes Reassignment
Get lists of HQ nodes and CN nodes assigned to US
Get current HQ QoS for US
Currrent HQ Qos < required HQ QoS?
Yes
Select new HQ nodes from HQ pool for restoring HQ missing quality
Unassign disconnected nodes from US and assign new HQ nodes to US
Update current HQ QoS for US
Update current US QoS
No
CN Nodes Lazy Reassignment
Is required US QoS met?
No
Get CN from CN nodes list assigned to US
Get current CN QoS
Yes
No
Current CN Qos < threshold T?
Yes
Replace failed nodes in CN node
Update CN QoS
Update current US QoS

Fig. 6. User service replicas reassignment algorithm flowchart.

Fig. 7 provides a brief sequence of the actions that occur after a node is disconnected. Starting from a scenario where HQ and CN nodes are both connected, and several USs replicas are hosted on them, two different types of events may occur; (1) a HQ node failure, or (2) a CN node failure.

In Fig. 7(a1), HQ node 3 disconnects and the US replicas kept in this node are no longer available. Since all nodes send heartbeat signals to the CCS, after some time without receiving them, the CCS will consider that node 3 is disconnected. The CCS will select new nodes to replicate the USs kept in node 3 (from users 2 and

Algorithm 2: User Service Replicas Reassignment Algorithm

```
Input: The minimum required HQ quality for a US: minUSHQRequiredQuality; The minimum required CN quality for a US: minUSCNRequiredQuality;
       The list of USs deployed in the network: UServices
Output: List of HQ nodes reassigned per US: USHQreassignedList; List of CN nodes reassigned per US: USCNreassignedList;
Procedure UserServiceReplicasReassignment:
    MinUSrequiredQuality ← MinUSHQrequiredQuality + MinUSCNrequiredQuality
    for each (UService in UServices) do
        USHQreassignedList, USCNreassignedList ← empty list
        (USHQnodesList, USHQquality) ← Get List of HQ Nodes and current US HQ quality from UService
        (USCNodesList, USCNquality) ← Get List of Complementary Nodes (CNs) and current US CN quality from UService
        if USHQquality < minUSHQrequiredQuality then
            missingUSHQquality ← minUSHQrequiredQuality − USHQquality
            USHQnewNodesList ← selectHQNodes(missingUSHQquality)
            Assign nodes to UService(USHQnewNodesList)
            USHQreassignedNodesList ← getDisconnectedNodes(USHQnodesList)
            Unassign nodes from UService(USHQreassignedNodesList)
            USHQquality ← Update US HQ Quality
            USHQreassignedList ← add (UService, USHQreassignedNodesList)
        end
        USquality ← USHQquality + USCNquality
        if USquality < MinUSrequiredQuality) then
            missingQuality ← minUSrequiredQuality − USquality
            (updatedCNquality, USCNreassignedNodesList) ← restoreCNquality(USCNodesList, missingQuality, UService)
            USquality ← USHQquality + updatedCNquality
            USCNreassignedList ← add (UService, USCNreassignedNodesList)
        end
    end
return USHQreassignedList,USCNreassignedList

Input: The list of CN nodes assigned to the user service: CNodesList; Missing quality of service required on CN nodes for the user service:
       missingCNQuality; User service: UService
Output: Updated quality on CN nodes after the replacement of nodes: updatedCNquality; List of CN nodes reassigned for restoring the missing
        quality: USCNreassignedNodesList
Procedure restoreCNQuality:
    USCNreassignedNodesList ← empty list
    while missingCNQuality > 0 do
        for each (CNode in CNodesList) do
            if CNode has all nodes disconnected then
                (newCN, newCNquality) ← Create new CN
                Unassign CNode from UService
                Assign CN to UService
                USCNreassignedNodesList ← getDisconnectedNodes(CNode)
            else
                CNFailedNodesList ← Get list of failed nodes of CNode
                for each (failedNode in CNFailedNodesList) do
                    failedNodeCluster ← Get cluster id of failedNode
                    (newNode, newNodeQuality) ← selectCNode(failedNodeCluster)
                    Unassign failedNode from CNode and UService
                    Assign newNode to CNode and UService
                    USCNreassignedNodesList ← failedNode
                end
                newCNquality ← Get quality from CNode
            end
            missingQuality ← missingQuality - newCNquality
        end
    end
    updatedCNQuality ← Get CN quality from CN nodes of UService
return updatedCNquality,USCNreassignedNodesList
```

3). The CCS decides that US from users 2 and 3 will go to nodes 4 and 2 respectively. Once nodes 4 and 2 are aware that they should host the USs from users 2 and 3, they ask for the node list currently hosting these USs (1) to the CCS (2). Then, nodes 2 and 4 select randomly a candidate node from the list and start a replication session with the selected nodes (3) and (4). Finally, Fig. 7(b1) shows the USs of users 2 and 3 replicated into nodes 4 and 2 respectively.

On the other hand, in Fig. 7(a2), one of the underlying LQ nodes in the CN node 1 is failed. The CCS evaluates the current CN node QoS, and in case it does not reach the minimum required CN QoS (threshold T), the CCS proceeds to replace the failed LQ nodes for restoring the missing CN QoS. The CCS unassigns the failed node and selects a new node to be added to the CN. Once the new node is aware that it has been added to a new CN, it asks for the list of LQ nodes in the CN (1) to the CCS (2). Then, the new node selects randomly a node from the LQ nodes list, and start a

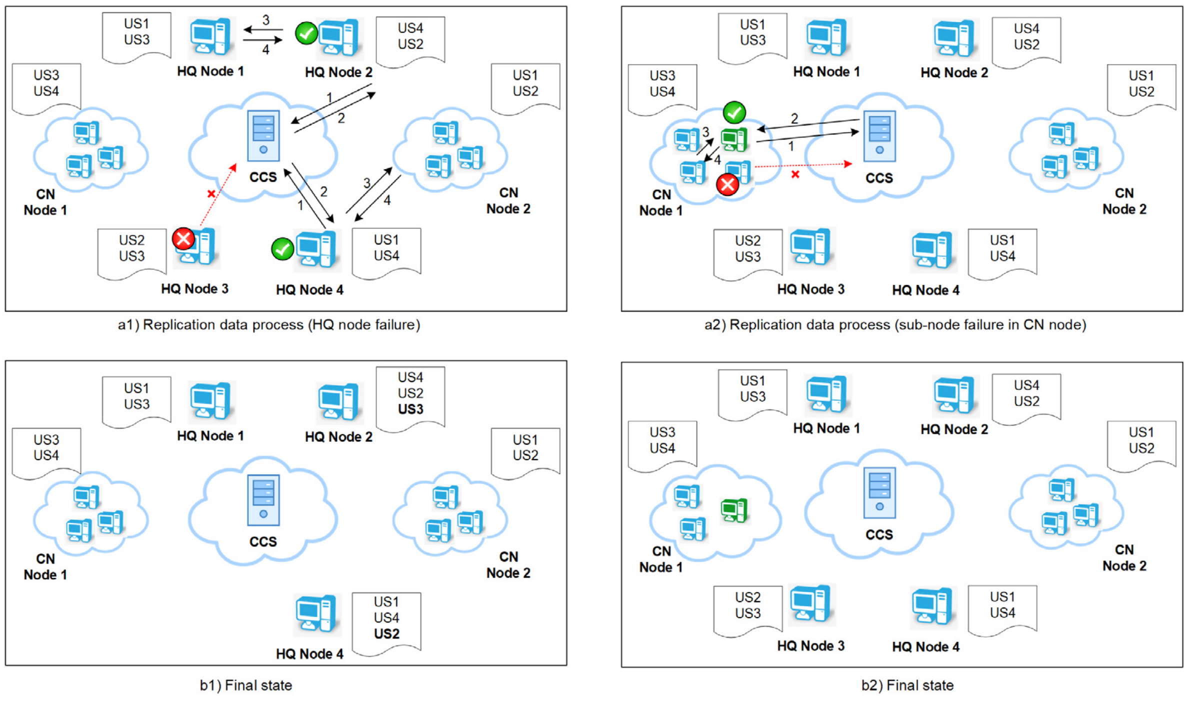


Fig. 7. User service replicas reassignment processes.

replication session with the selected node (3) and (4). All the USs hosted by the CN (user 3 and 4) are replicated in the new node. Once the replication is finished, the CCS updates the CN QoS with the new node. Finally, Fig. 7(b2) shows the new node hosting USs for users 3 and 4 and added to the CN.

5.3. Algorithms time complexity analysis

The algorithm 1 starts positioning on a cluster where a LQ node is selected from the sorted list of the cluster LQ nodes. Next, an iterative process is performed until complete the CN building. This process starts selecting the next cluster based on the Euclidean distance between the CN AP and each cluster centroid. Once the destination cluster has been selected, the algorithm selects the node with the highest Euclidean distance with the CN AP, and adds it to the CN, updating the correspondent CN AP and quality. This process is repeated until reaching a CN quality higher than the threshold T. Thus, the complexity of the algorithm depends on the number of clusters, the size of each cluster, and the length of the AP. The worst case would require iterating all the clusters and checking the Euclidean distance for all the nodes in the set of nodes C, with an AP length equal to $|H|$. Therefore, the algorithm 1 has a linear time complexity depending on the number of nodes ($|C|$) and the number of timeframes used in the AP ($|H|$). Concretely, $\mathcal{O}(|C| \cdot |H|)$. However, considering that the number of timeframes of the AP is constant ($|H|$=168 positions), we may conclude that the time complexity of algorithm 1 is $\mathcal{O}(|C|)$.

Regarding algorithm 2, the US replicas reassignment process aims to restore the missing US QoS. The USs are deployed in HQ and CN nodes which may get disconnected over time, and in consequence, the required US QoS may not be met. Then, it is needed to select new nodes for replicating the US data on them. The algorithm implements a lazy reassignment method that minimizes the number of nodes to be replaced, i.e. the minimum number of nodes required for restoring the missing US QoS. The asymptotic analysis exhibits that the worst case would require performing a reassignment of all the US replicas stored in the HQ and CN nodes assigned to the US, including on their underlying LQ nodes for each CN. Thus, the algorithm 2 has a linear time complexity depending on the number of USs, the number of nodes ($|C|$), and the number of timeframes of the AP ($|H|$). Concretely, $\mathcal{O}(|services| \cdot |C| \cdot |H|)$. However, given than the number of timeframes of the AP is constant ($|H|$ = 168 positions), we may conclude that the time complexity of algorithm 2 is $\mathcal{O}(|services| \cdot |C|)$.

6. Computational experiments

This section aims to describe the simulation environment and the experiments carried out to evaluate the goodness of the CLARA mechanism. As well, the clustering analysis and results obtained from the K-means clustering are detailed.

6.1. Failure Trace Archive

The nodes AP characterization is based on the analysis of historical availability node traces from the Failure Trace Archive (FTA)[2] [35,36], a public repository of traces of distributed systems with the purpose of facilitating the validation of fault-tolerant models and algorithms. This historical information represents the availability shown by about 230,000 nodes over the Internet from April 1, 2007, to January 1, 2009, on the usage of the SETI@Home VC application. The election of this historical information allows simulating nodes based on users' real behavior in a VC network, enabling the validation and verification process of the goodness of our mechanism, and the evaluation of the results that could be expected in a similar real network.

6.2. Simulation environment

Our mechanism has been validated in a real case scenario, using a real microblogging application named Garlanet. Garlanet is a Twitter-like decentralized implementation of a microblogging

[2] https://fta.scem.westernsydney.edu.au/.

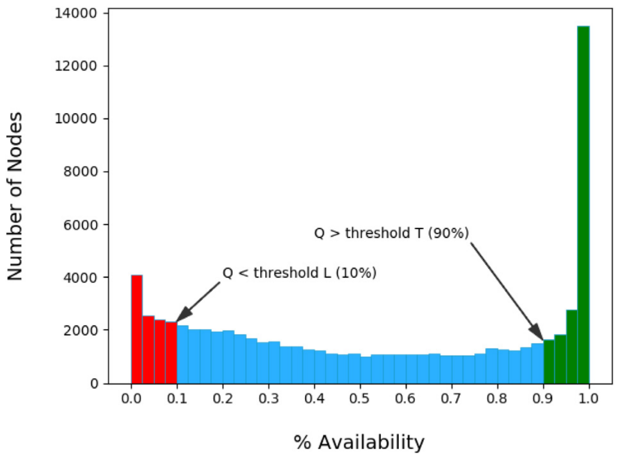


Fig. 8. Node qualities histogram before filtering (4-weeks sample interval).

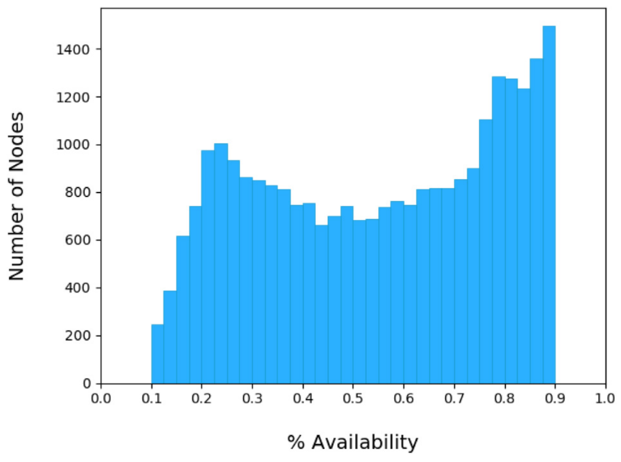


Fig. 9. Node qualities histogram after filtering (4-weeks sample interval).

social network, which stores and replicates the user data across different nodes to guarantee their availability. Garlanet has a CCS that oversees the status of the nodes, detecting which nodes are available at each moment, and assigning the most suitable nodes to clients. Moreover, the CCS guarantees that all the USs meet the constraints of a minimum number of replicas and required QoS. The simulator has been developed using Java 8 Standard Edition and aims to reproduce the behavior of this environment in the most realistic way. All the computational experiments have been carried out on a workstation with an Intel quad-core processor of 2.7 GHz with 8 GB of RAM memory. As an operating system, we have used Ubuntu 18.04.

6.3. Clustering results

The FTA trace allows us to build up the nodes AP for a period of 4 consecutive weeks randomly selected, according to the study performed by [34]. These nodes APs are later converted into node qualities following the process indicated in phase II. Table 3 shows the number of nodes loaded from the trace for a selected 4-weeks period, as well as the effectiveness of the filters. The histogram of node qualities before filtering for a prediction interval of 4 weeks (Fig. 8) exhibits a predominance of two main sets of nodes: very high-quality nodes and very low-quality nodes. To discard all the nodes exhibiting this very low-quality profile during the AP interval, a minimum required quality of 10% has been set for filter A.

Regarding the threshold T to split nodes into HQ and LQ nodes (filter B), an analysis of different values has been carried out. Table 3 summarizes the filtering results for T values equal to 90%, 85%, 80% and 75%. For selecting the T value, a clustering process has been performed using the K-means algorithm, observing that as the T value decreases, the clusters are splitted into smaller sub-clusters. Although these new sub-clusters allow building better CNs, the decrease of the T value leads to select less reliable HQ nodes, and therefore, the number of US replicas reassignments is increased, derived from the more frequent unavailability of these nodes initially categorized as HQ nodes. For this reason, our decision about the election of the threshold T has been guided by the minimization of the number of US replicas reassignments and the quality commitment with the users. To achieve both objectives, we have decided to use 90% as threshold T. This value will ensure to have nodes available practically most of the time while having very well-defined availability patterns (clusters) for building CNs.

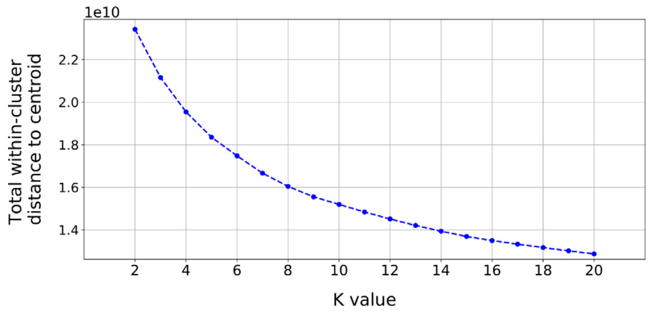


Fig. 10. Elbow plot for K-means clustering for K values between 2 and 20.

The filtering process results for a threshold of 90% shows a high predominance of nodes with a very-high and a very-low connection profile (around 42% of nodes), and four-fifths parts of nodes with a stable connection profile (21.67% of nodes do not exhibit an availability pattern on the AP interval). The filtering process has splitted the set of nodes into a 26.41% of HQ nodes and a 36.74% of LQ nodes, discarding the rest of the nodes. The histogram of node qualities, once the filters have been applied (Fig. 9), shows how all these sets of very-high and very-low quality nodes have been removed, and similar distributions of nodes along the remaining qualities are exhibited.

Next, we have grouped the LQ nodes set using the K-means clustering algorithm. Our mechanism considers the nodes APs as the set of observations to cluster. The AP is a two-dimensional vector of 168 positions (7 days x 24 h) for storing the number of minutes that nodes have been available at each timeframe during the AP interval. As a result, the nodes APs in the same cluster will be very similar between them, but quite different from the nodes APs on other clusters. To determine the most suitable K value, we have evaluated a wide enough range of K values. The total within-cluster distances have been represented in an elbow plot (Fig. 10) which exhibits how the overall cost decreases significantly as the number of K increases because large clusters are splitted into smaller clusters. The descent slope decreases more significantly for K values between 2 and 10, while greater values do not reduce considerably the overall cost.

Additionally, we have contrasted these results using the Dunn index for each cluster in order to determine the ratio between the maximum intra-cluster and the minimum inter-cluster distances. According to these results, we have concluded that K equal to 9 is the most appropriated value. Furthermore, we have considered other metrics to confirm this K value among all the

Table 3
Filtered nodes summary for different filter B thresholds.

Nodes in 4 weeks interval	Filter A Quality <10%)		Filter B (Quality >T%)			Filter C (Quality on S timeframes >T%)		Nodes respect initial num.
	Nodes after	Reduction	T %	Nodes after	Reduction	Nodes after	Reduction	
74,597	63,279	15.17%	90%	43,577	26.41%	27,411	21.67%	36.74% (27,411)
			85%	40,725	30.23%	24,559		32.91% (24,559)
			80%	38,217	33.59%	22,051		29.56% (22,051)
			75%	35,823	36.80%	19,663		26.35% (19,663)

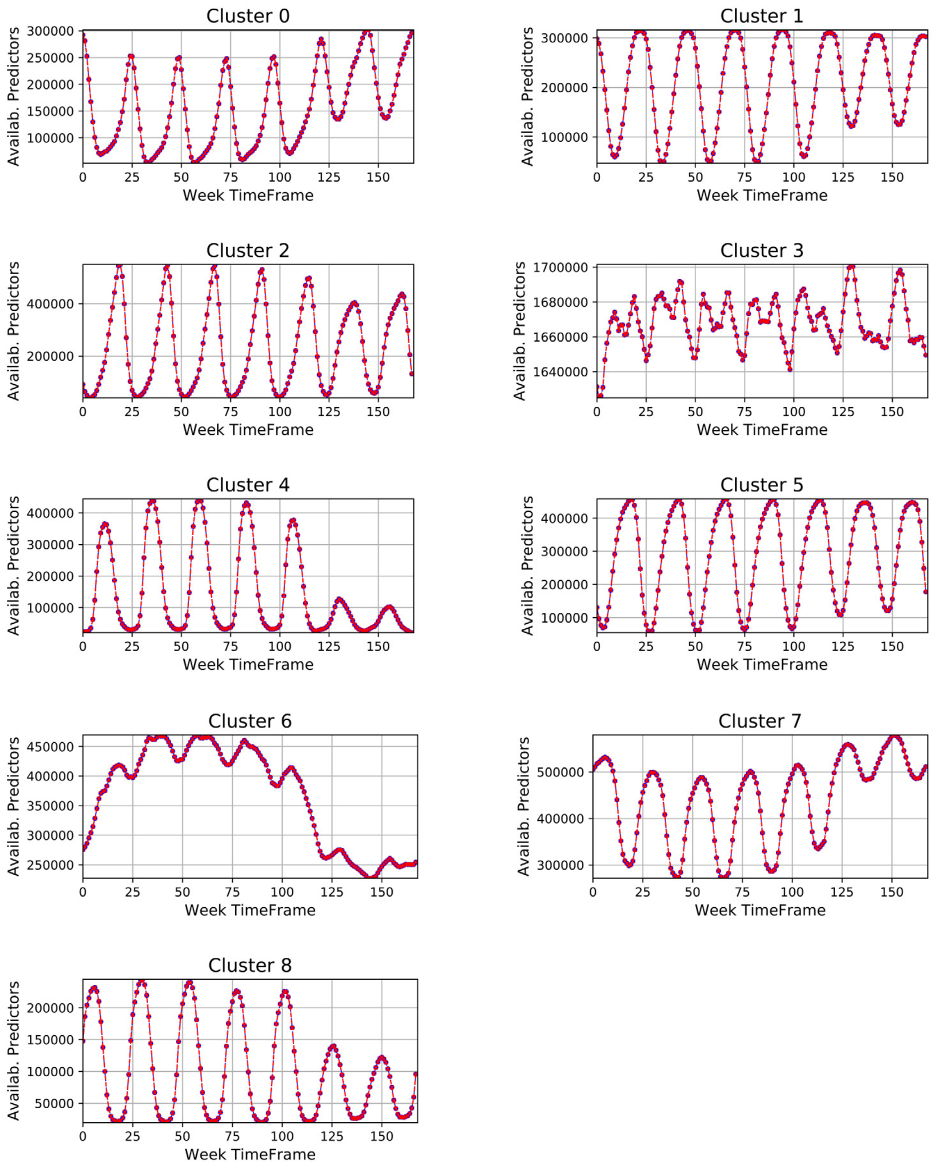


Fig. 11. K-Means - Bar Chart Plot for $K = 9$ and Threshold T=90%.

evaluated ones. On one hand, the Davies–Bouldin index measures the ratio between the within-cluster and the between-cluster distances, computing the average over all the clusters. On the other hand, the Calinski–Harabasz index compares the variance between clusters to the variance within each cluster. For both indices, K values between 6 and 11 are the best values, presenting slight differences in the indices' results.

Additionally, we have analyzed the bar chart diagrams for the promising K values for checking the fitting of the clusters in the AP patterns defined by its centroids (e.g., Fig. 11). These bar charts represent the sum of the cluster nodes APs, which will summarize the AP pattern defined by each cluster. The usage of several indices to determine the best K value has not been conclusive, possibly derived from the own nature of the clustered APs. However, the existence of slight differences over the indices has guided us to use a value of $K = 9$ instead of higher values. Additionally, the K-Means clustering has shown the existence of one cluster more populated than the others (cluster 3), which is not splitted into smaller and better-defined clusters as we increase the K value. The rest of the clusters exhibit a well-defined shape in terms of the AP pattern.

Table 4
Description of simulation scenarios.

	#USs	#nodes	HQ nodes	LQ nodes
Scenario 1	1,500	500	209	291
Scenario 2	3,000	1,000	411	589
Scenario 3	4,500	1,500	641	859
Scenario 4	6,000	2,000	840	1,160
Scenario 5	7,500	2,500	1,056	1,444
Scenario 6	9,000	3,000	1,263	1,737
Scenario 7	10,500	3,500	1,411	2,089
Scenario 8	12,000	4,000	1,709	2,291

6.4. Simulation scenarios

The purpose of the simulation phase is to validate the goodness of the CLARA mechanism as a RA method for USs placement. The simulation scenarios aim to establish a comparison framework between our mechanism and a RA method only based on the usage of the most available nodes, or alternatively, a RA method that gives more preference to them.

In this sense, we have defined a comparison analysis with the method proposed by [29]. This method prioritizes the most available nodes for SP through the definition of a Multi-Criteria Biased Randomized (MCBR) mechanism. The MCBR method is a hierarchical method based on a Lexicographic Ordering (LO) multicriteria optimization strategy in which the intrinsic properties of the nodes are categorized into different priority levels. Then, a sequence of sub-optimization problems is solved following the previously established order of priority. Finally, a reduced list of suitable nodes for deploying the USs is obtained, which represents the sorted list of nodes with the highest quality. To select the final node to use, biased randomization techniques are used to distribute the load among the HQ selected nodes.

The impossibility of simulating all the nodes in the 4-weeks AP interval (see Table 3), because of the required computing capacity, has enforced us to select lower workflows based on a random selection of nodes from the 4-weeks set of nodes. This selection process validates that the distribution of nodes in the new subset follows the same distribution as the original set (see Table 4). The simulator executes several runs using these workflows on both methods for comparing the results.

Finally, the distribution percentage of qualities (α parameter) between the HQ and CN nodes is also analyzed in order to study the USs sensitivity on the resulting metrics.

7. Analysis of results

The simulation results aim to provide information about the goodness of the CLARA mechanism under different areas of study. These analyses include the behavior of the mechanism under different qualities distributions (α parameter) in comparison with the MCBR-based RA method described in [29].

Relative to the **US replication metrics** (Table 5), the average number of available replicas of a US remains stable on practically all the evaluated α values, achieving for $\alpha = 50\%$ similar values to the ones obtained for the MCBR-based RA method (9.59 vs 9.65 replicas per timeframe). The CLARA mechanism spreads USs on more nodes in the network, reducing the overhead on the HQ nodes, and increasing the capacity of the network without impacting the US replicas available. Moreover, all the USs practically exhibit available replicas all the time which leads to a continuous service availability. Only in a very few cases, the US is provided in a degraded mode due to the unavailability of replicas on HQ and CN nodes.

Regarding the **US replicas reassignment metrics** (Table 6), our mechanism exhibits a significant drop on the number of

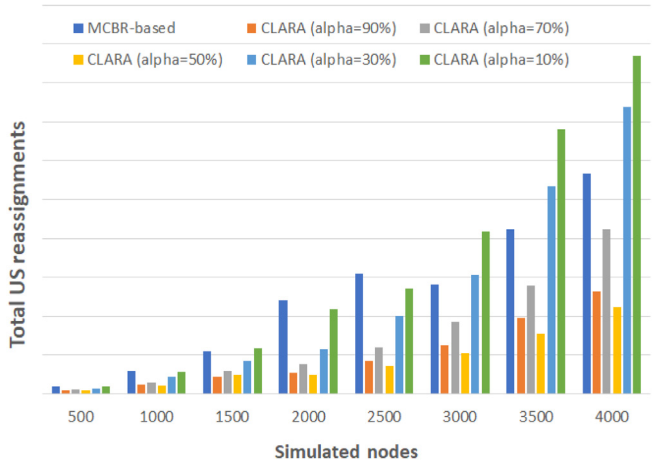


Fig. 12. Total US replicas reassignments comparison.

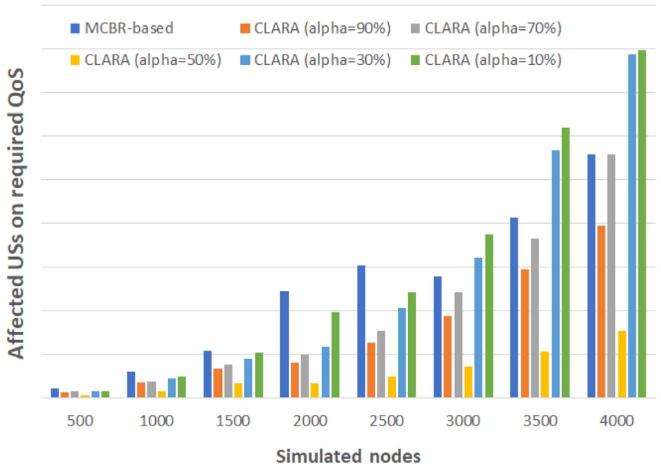


Fig. 13. USs impacted on required QoS before US replicas reassignment comparison.

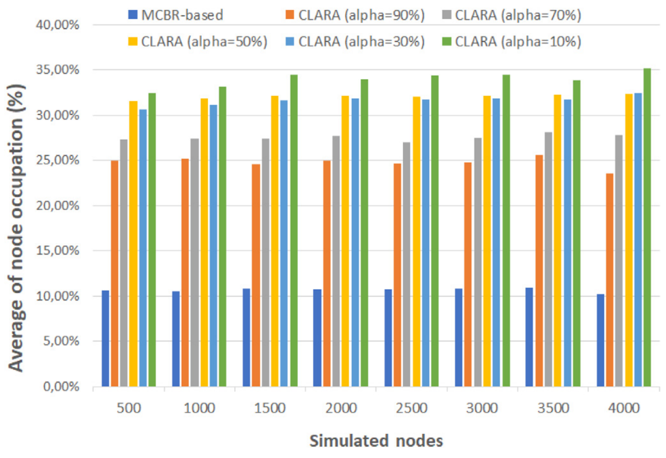


Fig. 14. Average on node occupation comparison.

US replicas reassignments, in comparison to the MCBR-based RA method, for α values greater than 50% (see Fig. 12), achieving the lowest figures for $\alpha = 50\%$ where the total decrease is around the 64%. As the required US QoS depends more on CNs than HQ nodes (α values less than 50%), the number of US replicas reassignments is remarkably increased which also leads, in consequence, to

Table 5
US replication results comparison (scenario 6).

	MCBR-based	CLARA				
		$\alpha = 90\%$	$\alpha = 70\%$	$\alpha = 50\%$	$\alpha = 30\%$	$\alpha = 10\%$
US without replicas available at a timeframe	0	0	0	0	0	0
US without HQ replicas available at a timeframe	0	0	0	5.29E−7	1.231E−4	0.0382
US without CN replicas available at a timeframe	0	0.013	4.61E−4	3.96E−7	1.32E−7	0
Avg. US replicas per timeframe	9.65	8.85	8.80	9.59	8.78	8.69
Avg. US replicas per timeframe on HQs	9.65	7.86	5.85	4.89	2.94	0.96
Avg. US replicas per timeframe on CNs	0	0.99	2.95	4.70	5.84	7.73

Table 6
US replicas reassignments results comparison (scenario 6).

	MCBR-based	CLARA				
		$\alpha = 90\%$	$\alpha = 70\%$	$\alpha = 50\%$	$\alpha = 30\%$	$\alpha = 10\%$
US replicas reassignments on HQs	2,817,853	1,080,801	1,096,644	823,355	749,344	288,853
US replicas reassignments on CNs	0	180,007	757,063	212,439	2,318,177	3,876,985
HQs not available at US replicas reassignments	10	12	22	6	6	10
CNs not available at US replicas reassignments	0	15	340	136	4,187	15,813
Unserved USs by absence of HQs (Q_{USHQ}=0)	0	0	0	4	931	288,863
Unserved USs by absence of CNs (Q_{USCN}=0)	0	103,992	348	3	1	0
US replicas reassignments over complete CNs	0	67,383	294,202	84,708	913,356	1,540,150
USs impacted on Q_{US} before US replicas reassignments	1,386,087	934,333	1,211,020	356,642	1,605,864	1,864,507
USs impacted on Q_{US} after US replicas reassignments	10	22	254	72	2,536	8,594
USs impacted on Q_{USHQ} before US replicas reassignments	1,386,087	871,966	938,721	726,711	704,263	288,863
USs impacted on Q_{USHQ} after US replicas reassignments	10	10	20	6	6	10
USs impacted on Q_{USCN} before US replicas reassignments	0	193,061	438,379	2,153,214	1,357,271	1,730,236
USs impacted on Q_{USCN} after US replicas reassignments	0	89,131	39,773	2,146,594	292,597	70,706

increase the number of USs affected on the required QoS (see Fig. 13). However, a significant reduction is exhibited for $\alpha = 50\%$ respecting the MCBR-based RA method, which represents a reduction of 75% over the affected USs. In this case, the number of unrestored USs QoS is practically identical for both $\alpha = 50\%$ and MCBR-based RA method, being slightly higher for $\alpha = 50\%$ (10 vs 72). For $\alpha = 50\%$, almost all the US replicas reassignments have been performed on HQ nodes, being only necessary to restore the 0.003% of USs (6,620 of a total of 2,153,214 USs affected by the CN quality) to restore the required US QoS.

Finally, the number of US replicas reassignments that could not be completed by the absence of available nodes is higher on all scenarios respecting the MCBR-based RA method. For $\alpha = 50\%$, a total of 142 (6 HQ + 136 CN) US replicas reassignments were unperformed while only 9 for the MCBR-based RA method. Comparing all the evaluated α values, the number of total unperformed US replicas reassignments remains stable for α values between 90% and 50%, increasing dramatically for $\alpha = 30\%$ and $\alpha = 10\%$ as a consequence of difficulties for finding free nodes in the clusters for the USs reallocation. Thereby, as the US quality depends more on CNs (low values of α% parameter), the cluster's nodes become scarcer, making it more difficult to create effective CNs. However, for α values equal to or greater than 50%, the number of unperformed US replicas reassignments may be considered negligible compared with the total number of US replicas reassignments performed.

Regarding the **capacity metrics** (Table 7), there is an increase in the resources consumption for all the evaluated α values compared with the MCBR-based RA method as a consequence of a higher replication on more nodes (CNs). As well, the number of nodes that are full of capacity increases as the required US QoS depends more on CNs than on HQ nodes (see Fig. 14). The average occupation increases with respect to MCBR-based RA method is around 20%.

However, this capacity increment exhibited in the node's storage was expected because the number of nodes involved in the SP is higher on our mechanism. However, the spreading of USs on more nodes has two main advantages that are confirmed with the simulation results. On one hand, the HQ nodes are less overloaded with our mechanism. For $\alpha = 50\%$, only the 0.20% of the HQ nodes are full of capacity while the 6.70% of nodes are full of capacity for the MCBR-based RA method. On the other hand, there is a wider usage of the network resources while the required US QoS requirements are satisfied.

Finally, Table 8 shows a execution times comparison on the allocation of resources for a single US. The CLARA mechanism is not designed to deploy a set of USs "all at once". Instead, CLARA allows taking a decision about the resources to be allocated when a new US is required for placement. Thus, this comparison aims to evaluate the execution times required for deciding the list of nodes for a new US placement.

The results shows that the mean value for allocating resources for a single US, using the MCBR-based RA method, is 1 ms versus 15 ms for the worst CLARA scenario (highest US dependency on CNs). However, in return, CLARA leverages resources that otherwise would have been discarded. The CN creation process has an impact on the RA time because of the clusters iteration for finding suitable nodes while building up new CNs. Thus, as the required US quality depends more on CNs (α lower than 50%), the impact on the execution time will be higher given than more CNs will be needed for meeting the required US QoS. Thus, the CN creation process introduces an additional cost in time, respecting the MCBR-based RA method, that depends on the distribution of US qualities (α value), the number of clusters, and the size of the clusters.

Additionally, Fig. 15 shows the execution and CPU consumption times spent on US reassignments throughout a simulated week, showing how these times are distributed over time. The US reassignment process is performed five times per timeframe, which is equivalent to a US reassignment slot every 12 min on the simulated time. Green data represents the time spent on each US reassigned. The orange data represents the sum of times of all the reassigned USs on each slot i.e. the total reassignment time spent in the slot. Finally, the blue line represents the mean value of the sum of all these total reassignment times.

The execution and CPU times for US reassignments on CNs are slightly greater than on HQs because of clusters iteration for finding suitable replacement nodes. The comparison of execution

Table 7
Capacity results comparison (scenario 6).

	MCBR-based	CLARA				
		α = 90%	α = 70%	α = 50%	α = 30%	α = 10%
HQs full of capacity (#)	200	64	5	2	0	0
HQs full of capacity (%)	6.70%	5.10%	0.40%	0.20%	0%	0%
CNs full of capacity (#)	0	16	77	113	203	279
CNs full of capacity (%)	0%	0.90%	4.40%	6.80%	11.70%	16.10%
Avg. usage respect USs deployed on HQs	10.90%	20.80%	15.40%	13%	7.20%	2.50%
Avg. usage respect USs deployed on CNs	0%	4%	12.10%	19.20%	24%	32%

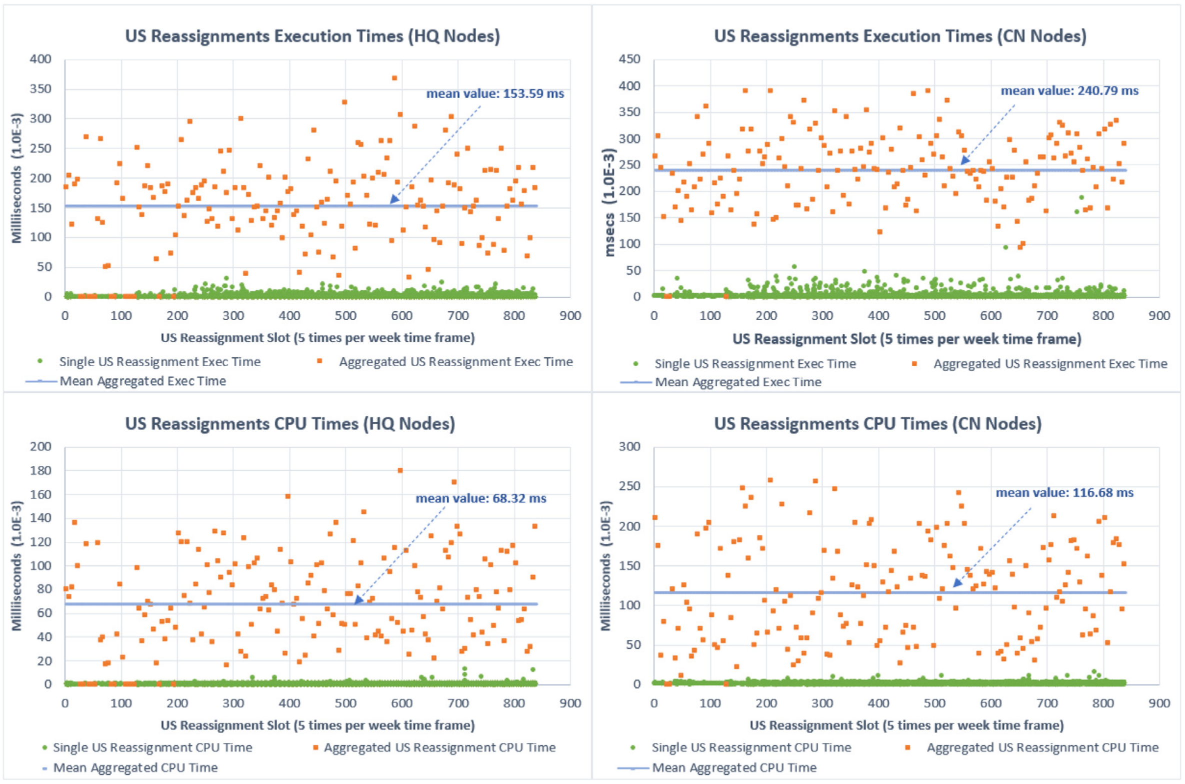


Fig. 15. Comparison of CPU and execution times of US reassignments (scenario 3).

Table 8
Comparison of the execution time when allocating a US (scenario 6).

	mean (msecs)	std dev (msecs)
MCBR-based	1.02	0.61
CLARA (alpha=90%)	3.19	2.31
CLARA (alpha=70%)	7.11	3.83
CLARA (alpha=50%)	10.71	5.05
CLARA (alpha=30%)	12.28	5.75
CLARA (alpha=10%)	15.49	6.29

times for HQs mostly exhibits values in the range of 50 to 300 ms versus values in the range of 150 to 400 ms for CNs. Respecting the CPU time, HQs exhibits values in the range of 20 to 140 ms versus values in the range of 30 to 250 ms for CNs. In summary, the CLARA mechanism exhibits an increase on the execution time of 87 ms on average and an increase on the CPU time of 48 ms on average over the scenario with 4,500 US deployed (scenario 3).

7.1. Statistical hypothesis testing

The tests are based on the Wilcoxon signed-rank test. This test is a non-parametric statistical hypothesis test commonly used to compare two related samples, solutions of two algorithms in our case. In particular, this test aims to assess whether their population mean ranks differ. A confidence level of 95% is set. The programming language R (version 4.1.0) (R Core Team, 2017) has been used to carry out the hypothesis testing. The tests performed cover the main metrics used to validate the goodness of the mechanism, which are related to the number of replicas used by the USs, the usage of the nodes respecting the number of US deployed in the network, and the number of US replicas reassignments.

The Wilcoxon signed-rank test results and their respective conclusions are the following:

- Average US replicas per timeframe: Test statistic W=32.5; p-value=1; Conclusion: we do not reject H_0. We do not have statistically significant evidence to show that the difference between mean ranks in the metric is not zero.
- US replicas reassignments:Test statistic W=53; p-value = 0.0281; Conclusion: we do reject H_0. We do have statistically significant evidence to show that the difference between mean ranks in the metric is not zero.
- Average usage respect USs deployed: Test statistic W=32.5; p-value=0.0009; Conclusion: we do reject H_0. We do have statistically significant evidence to show that the difference between mean ranks in the metric is not zero.

8. Conclusions and future research

In this paper, we present the CLARA mechanism, a novel clustering-based RA method that leverages donated resources exhibiting a low-availability profile, maximizing the eligible set of resources for USs deployment. As result, both nodes exhibiting high and low availability profiles are considered by the RA mechanism for SP. Unlike most of the existing RA methods in VC networks where the low-available nodes are often discarded, CLARA considers nodes exhibiting low availability patterns for smartly combining them into groups based on their complementary availability relationship. To speed up the process for building up effective CNs, the CLARA mechanism relies on clustering techniques to group nodes into disjoint sets based on their AP profile. As result, the capacity of the network for providing USs is highly increased while the overloading of the HQ nodes is remarkably reduced.

The analysis of the results exhibits how our mechanism maximizes the use of the poor quality computational resources to satisfy the required US QoS while minimizes the costly number of US replicas reassignments between nodes. The creation of CNs jointly with the lazy reassignment algorithm show how LQ nodes may work over time in a combined way to exhibit an availability level equivalent to a HQ node. Furthermore, our mechanism increases the set of eligible resources for SP while avoids the overloading of the most available nodes.

Future research lines are focused on the clustering process. The K-means algorithm requires selecting the K value which may evolve over time as nodes availability patterns change, so it requires a periodic verification process inside the CCS to re-evaluate the most suitable K value. Moreover, additional smarter aspects may be incorporated into the CNs creation algorithm for increasing the reliability of the set and be able to define different QoS for the USs. Thus, node parameters such as connection speed, geographical location, or level of occupancy among others could be incorporated into the algorithm. As well, extensions on the US replicas reassignment algorithm must be analyzed in order to perform a selection of nodes based on replication cost, bandwidth, processing time, and node location to minimize costs and recovery times for USs temporarily running in a degraded mode (required US QoS not achieved).

CRediT authorship contribution statement

Sergio Gonzalo: Formal analysis, Investigation, Methodology, Software, Visualization, Roles/Writing – original draft. **Joan Manuel Marquès:** Conceptualization, Methodology, Supervision, Validation, Writing – review & editing. **Alberto García-Villoria:** Formal analysis, Supervision, Writing – review & editing. **Javier Panadero:** Methodology, Software, Validation, Writing – review & editing. **Laura Calvet:** Formal analysis, Methodology, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science, Innovation and Universities (PGC2018-097599-B-100).

References

[1] M.N. Durrani, J.A. Shamsi, Volunteer computing: requirements, challenges, and solutions, J. Netw. Comput. Appl. 39 (2014) 369–380.
[2] T.M. Mengistu, D. Che, Survey and taxonomy of volunteer computing, ACM Comput. Surv. 52 (3) (2019) 1–35.
[3] M. Selimi, F. Freitag, Towards application deployment in community network clouds, in: International Conference on Computational Science and Its Applications, Springer, 2014, pp. 614–627.
[4] D.L. Iglesias, J.-M. Marquès, G. Cabrera, H. Rifa-Pous, A. Montane, HorNet: microblogging for a contributory social network, IEEE Internet Comput. 16 (3) (2012) 37–45.
[5] S. Oukemeni, H. Rifa-Pous, J.M. Marquès, Privacy analysis on microblogging online social networks: a survey, ACM Comput. Surv. 52 (3) (2019) 1–36.
[6] A. Juan, J. Faulin, S. Grasman, M. Rabe, G. Figueira, A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems, Oper. Res. Perspect. 2 (2015) 62–72.
[7] X. Serra, J. de Armas, J.M. Marquès, Simulating and optimizing resource allocation in a micro-blogging applicacion, in: 2016 Winter Simulation Conference (WSC), IEEE, 2016, pp. 3167–3176.
[8] J. Panadero, L. Calvet, J.M. Marquès, A.A. Juan, A simheuristic approach for resource allocation in volunteer computing, in: 2017 Winter Simulation Conference (WSC), IEEE, 2017, pp. 1479–1490.
[9] T.C. Xavier, I.L. Santos, F.C. Delicato, P.F. Pires, M.P. Alves, T.S. Calmon, A.C. Oliveira, C.L. Amorim, Collaborative resource allocation for cloud of things systems, J. Netw. Comput. Appl. 159 (2020) 102592.
[10] T. Mengistu, D. Che, S. Lu, Multi-objective resource mapping and allocation for volunteer cloud computing, in: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), IEEE, 2019, pp. 344–348.
[11] Y. Alsenani, G. Crosby, T. Velasco, A. Alahmadi, Remot reputation and resource-based model to estimate the reliability of the host machines in volunteer cloud environment, in: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud, IEEE, 2018, pp. 63–70.
[12] Y. Alsenani, G. Crosby, T. Velasco, Sara: A stochastic model to estimate reliability of edge resources in volunteer cloud, in: 2018 IEEE International Conference on Edge Computing, IEEE, 2018, pp. 121–124.
[13] G. Rjoub, J. Bentahar, O.A. Wahab, Bigtrustscheduling: Trust-aware big data task scheduling approach in cloud computing environments, Future Gener. Comput. Syst. 110 (2020) 1079–1097.
[14] E. Sherzer, H. Levy, Resource allocation in the cloud with unreliable resources, Perform. Eval. 137 (2020) 102069.
[15] J. Zhang, X. Yang, N. Xie, X. Zhang, A.V. Vasilakos, W. Li, An online auction mechanism for time-varying multidimensional resource allocation in clouds, Future Gener. Comput. Syst. 111 (2020) 27–38.
[16] T. Ghafarian, B. Javadi, Cloud-aware data intensive workflow scheduling on volunteer computing systems, Future Gener. Comput. Syst. 51 (2015) 87–97.
[17] K. Xiao, Z. Gao, W. Shi, X. Qiu, Y. Yang, L. Rui, EdgeABC: An architecture for task offloading and resource allocation in the internet of things, Future Gener. Comput. Syst. 107 (2020) 498–508.
[18] I. Al Ridhawi, M. Aloqaily, Y. Jararweh, An incentive-based mechanism for volunteer computing using blockchain, ACM Trans. Internet Technol. 21 (4) (2021) 1–22.
[19] G.E. Gonçalves, P.T. Endo, M. Rodrigues, D.H. Sadok, J. Kelner, C. Curescu, Resource allocation based on redundancy models for high availability cloud, Computing 102 (1) (2020) 43–63.
[20] X. Chen, H. Wang, Y. Ma, X. Zheng, L. Guo, Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model, Future Gener. Comput. Syst. 105 (2020) 287–296.
[21] B. Ali, M.A. Pasha, S. ul Islam, H. Song, R. Buyya, A volunteer-supported fog computing environment for delay-sensitive IoT applications, IEEE Internet Things J. 8 (5) (2020) 3822–3830.
[22] C. Bayliss, J. Panadero, L. Calvet, J.M. Marquès, Reliability in volunteer computing micro-blogging services, Future Gener. Comput. Syst. 115 (2021) 857–871.
[23] C. Bayliss, J. Panadero, L. Calvet, J.M. Marquès, A simulation model for volunteer computing micro-blogging services, in: 2020 Winter Simulation Conference (WSC), IEEE, 2020, pp. 552–562.
[24] A. Waheed, M.A. Shah, A. Khan, C. Maple, I. Ullah, Hybrid task coordination using multi-hop communication in volunteer computing-based VANETs, Sensors 21 (8) (2021) 2718.
[25] W.M. Danquah, D.T. Altilar, Vehicular volunteer computing (VVC): A novel paradigm for the volunteering of vehicular resources; opportunities and challenges, in: 2019 International Conference on Computing, Electronics & Communications Engineering (ICCECE), IEEE, 2019, pp. 208–213.
[26] F. Zeng, Q. Chen, L. Meng, J. Wu, Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (6) (2021) 3247–3257.
[27] M. Ghobaei-Arani, A workload clustering based resource provisioning mechanism using biogeography based optimization technique in the cloud based systems, Soft Comput. 25 (5) (2021) 3813–3830.

[28] A. Shahidinejad, M. Ghobaei-Arani, M. Masdari, Resource provisioning using workload clustering in cloud computing environment: a hybrid approach, Cluster Comput. 24 (1) (2021) 319–342.

[29] J. Panadero, J. de Armas, X. Serra, J.M. Marquès, Multi criteria biased randomized method for resource allocation in distributed systems: Application in a volunteer computing system, Future Gener. Comput. Syst. 82 (2018) 29–40.

[30] M. Selimi, L. Cerdà, M. Sánchez-Artigas, F. Freitag, L. Veiga, Practical service placement approach for microservices architecture, in: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE, 2017, pp. 401–410.

[31] M. Selimi, L. Cerdà Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, J. Crowcroft, A lightweight service placement approach for community network micro-clouds, J. Grid Comput. 17 (1) (2019) 169–189.

[32] T.M. Mengistu, D. Che, A. Alahmadi, S. Lu, Semi-markov process based reliability and availability prediction for volunteer cloud systems, in: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), IEEE, 2018, pp. 359–366.

[33] O.A. Wahab, N. Kara, C. Edstrom, Y. Lemieux, MAPLE: A machine learning approach for efficient placement and adjustment of virtual network functions, J. Netw. Comput. Appl. 142 (2019) 37–50.

[34] D. Lázaro, D. Kondo, J.M. Marquès, Long-term availability prediction for groups of volunteer resources, J. Parallel Distrib. Comput. 72 (2) (2012) 281–296.

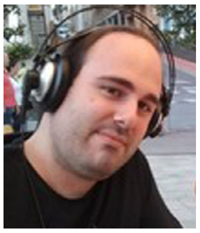
[35] D. Kondo, B. Javadi, A. Iosup, D. Epema, The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems, in: 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE, 2010, pp. 398–407.

[36] B. Javadi, D. Kondo, A. Iosup, D. Epema, The failure trace archive: Enabling the comparison of failure measurements and models of distributed systems, J. Parallel Distrib. Comput. 73 (8) (2013) 1208–1223.

Sergio Gonzalo is a Ph.D. candidate in the Internet Interdisciplinary Institute at the Universitat Oberta de Catalunya (UOC), Spain. He graduated as Telecommunications Engineer from the Technical University of Madrid (UPM) and holds a Master of Science (MsC) on Telecommunications Engineering from Universitat Oberta de Catalunya (UOC) and Universitat Ramon Llull (URL). His research interests include availability-aware distributed mechanisms for sets of non-dedicated resources, distributed computing, and networking algorithms.

Joan Manuel Marquès is an Associate Professor at Computer Science, Multimedia & Telecommunication Studies at Universitat Oberta de Catalunya since 1997. He graduated as a Computer Science Engineer from the Facultat d'Informàtica de Barcelona (UPC) and a Ph.D. in Computer Science from UPC. His research interests include the design of scalable and cooperative Internet services and applications. He is member of the Association for Computing Machinery.

Alberto García-Villoria is an Associate Professor at the Management Department at Universitat Politècnica de Catalunya (UPC) and a Course instructor at Universitat Oberta de Catalunya (UOC) He graduated as Computer Science Engineer and a Ph.D. in Advanced Automation and Robotics, both from UPC. His research expertise includes the design and application of operation research techniques to solve optimization problems. He has co-authored 38 papers published in JCR journals.



Javier Panadero is a PostDoc researcher at the Computer Science, Multimedia and Telecommunication Department at Open University of Catalonia (UOC). His major research areas are: performance prediction of HPC applications, Modeling and analysis of parallel applications and Simulation of parallel and distributed system. He has co-authored a total of 10 full-reviewed technical papers in journals and conference proceedings.

Laura Calvet is an Assistant Professor of Statistics at the Computer Science, Multimedia \& Telecommunication Studies at the Universitat Oberta de Catalunya. Her main lines of research are: design of optimization algorithms relying on metaheuristics, machine learning and/or simulation applied to sustainable logistics \& computing; and applied statistics \& economics.