Received: 14 November 2019 | Revised: 8 January 2020 | Accepted: 8 February 2020

DOI: 10.1002/ett.3924

SPECIAL ISSUE ARTICLE

WILEY

A comparative node evaluation model for highly heterogeneous massive-scale Internet of Things-Mist networks

Amin Shahraki[1,2] | **Marius Geitle**[1,2] | **Øystein Haugen**[1]

[1]Faculty of Computer Sciences, Østfold University College, Halden, Norway

[2]Department of Informatics, University of Oslo, Oslo, Norway

Correspondence
Amin Shahraki, Department of Informatics, University of Oslo, Gaustadalléen 23B, Oslo 0373, Norway.
Email:am.shahraki@ieee.org

Abstract

Internet of Things (IoT) is a new technology that is driving the connection of billions of devices around the world. Because these devices are often resource-constrained and very heterogeneous, this presents unique challenges. To address some of these challenges, new paradigms of Edge and Fog are emerging to bring computational resources of the IoT networks from remote devices like cloud closer to the end-devices. Mist computing is a new paradigm that attempts to make use of the more resource-rich nodes that are closer than Edge nodes to end-users. Since these nodes might have enough resources to host services, execute tasks or even run containers, the utilization of network resources might be improved, and delay reduced by utilizing these nodes. The nodes must, therefore, be assessed to determine which nodes should offer resources to other nodes based on their situation. In this article, a new comparative assessment model for ranking Mist nodes in highly heterogeneous massive-scale IoT networks in order to discover nodes that can offer their resources is proposed. The Mist nodes are evaluated based on parameters like resources, connections, applications, and environmental parameters to heuristically compare the neighbors with a novel learning-to-rank method to predict a suitability score for each node. The most suitable neighbor is then selected based on the score, with load balancing accomplished by a second chance method. When evaluating the performance, the results show that the proposed method succeeds in identifying resource-rich nodes, while considering the selection of other nodes.

1 | INTRODUCTION

The Internet of Things (IoT) is recognized as the most important emerging network infrastructure to connect billions of devices around the world.[1] It is increasingly set to become a vital aspect of technology, enabling "things" as devices to sense, communicate, and perform actions. Recently, new applications and developments in IoT have heightened the need for improving the efficiency of the network infrastructure to satisfy a wide range of requirements, such as reducing the energy consumption of the nodes and supporting quality of service (QoS) of different applications.[2] In addition,

the rapid expansion in the use of IoT in a diverse range of application areas means the current technology unable to maintain the efficiency of IoT networks regarding the huge volume of data to process and number of nodes to manage. There are increasing concerns that some IoT networks are being disadvantaged because of their massive-scale size and QoS problems.[3] In addition, the underutilization of resources and latency in accessing rich remote resources like the cloud are significant challenges for the current IoT networks.[4] Recently, researchers have been attempting to solve these problems by migrating computing capabilities from Far-end nodes like the cloud to so-called Edge nodes, which are geographically closer to the end-users.[5] The concept of Edge computing is a solution to reduce latency when accessing computational resources at Far-end Nodes and also to improve resource consumption by eliminating the need to connect to remote service in some cases. Edge nodes are generally part of the network infrastructure and are resource-rich nodes like routers, access points, base stations, or gateways. There are currently several different edge computing paradigms that are emerging simultaneously; namely, multiaccess edge computing,[6] mobile cloud computing[7] Fog computing,[8] and Mist computing.[9] While each of these paradigms is concerned with leveraging the computational resources close to end-users in order to perform computation tasks, they achieve this in different layers of the network. Because there are several layers available, its not clear which layer is best utilized to achieve these goals in different situations.[10]

When comparing the Edge computing paradigms, the research to date has focused on layers in the network infrastructure devices (Edge and Fog) to execute tasks, rather than resource-rich end-user devices. Although network infrastructure devices are appropriate to execute tasks and host services, there are reasons to encourage the pushing of execution closer to the end-users:

- IoT devices are becoming more powerful than before, which enables them to offer free resources to various light-weight or even heavy-weight tasks, in order to help their neighbors and the local network to improve performance.[11,12]
- In some IoT applications, the network load consumes a considerable amount of energy and bandwidth to transfer data to the edge of the network. Although Edge nodes are close to the end-users, moving the services even closer to end-users can reduce the network load, leading to more efficient resource utilization.[13]
- Moving the location of specific data processing models like caching, or data fusion, closer to end-users can exponentially improve the efficiency of a network by decreasing the amount of data being transmitted across the network.[14]
- In IoT networks, there are almost always free resources on IoT nodes that could be used to help the network to improve efficiency. In massive-scale IoT (mIoT) networks, the amount of these free resources can be considerable. Utilizing these can remove the costs of using remote resources (like the cloud) and network infrastructure resources like bandwidth.[15]
- Combining the free resources of weak devices can provide a resource-dedicated distributed network infrastructure to execute heavy computation tasks, which will help to save resources and improve the efficiency of the network.[16]

Mist computing is a newly defined paradigm that moves computing tasks from the cloud to resource-rich IoT devices; specifically, end-users like home appliances, and resource-rich sensors.[17] Compared with other paradigms of Edge computing, at the moment, Mist computing is commonly used because of the predominance of weak and resource-constrained end-user nodes, when compared with Edge nodes and the complexity of network management in a network with a considerable number of nodes.[18]

Although the literature review did not discover any publications that surveyed how quickly the resources of IoT nodes are increasing, it is evident that IoT nodes are becoming more powerful thanks to advancements in digital electronics. Today, wearable sensors are equipped with several network interfaces and have their own operating systems, which enable them to execute tasks, both light and heavy. Thanks to affordable modules and processors, like Raspberry Pi, Arduino, and ARM, most IoT nodes are capable of executing tasks locally without the help of the cloud. Evidence shows that the future of IoT networks belongs to the network paradigms that can use and manage the available network resources efficiently in all layers, because of the volume of data that needs to be processed.[19] Although providing resources on the edge of the network is a solution, there is no logical reason to ignore the free resources of IoT nodes that are more accessible, free, and flexible.[10]

Despite the efficient promise of Mist computing, it still suffers from several significant drawbacks. There is not yet much research on implementing Mist computing and a lack of standards for how to do it. How to implement Mist is made complex by the number of heterogeneous nodes found in mIoT networks, and there is an ongoing debate about the best strategies to follow, as they are not as easy to configure efficiently as, for example, Edge paradigms that generally

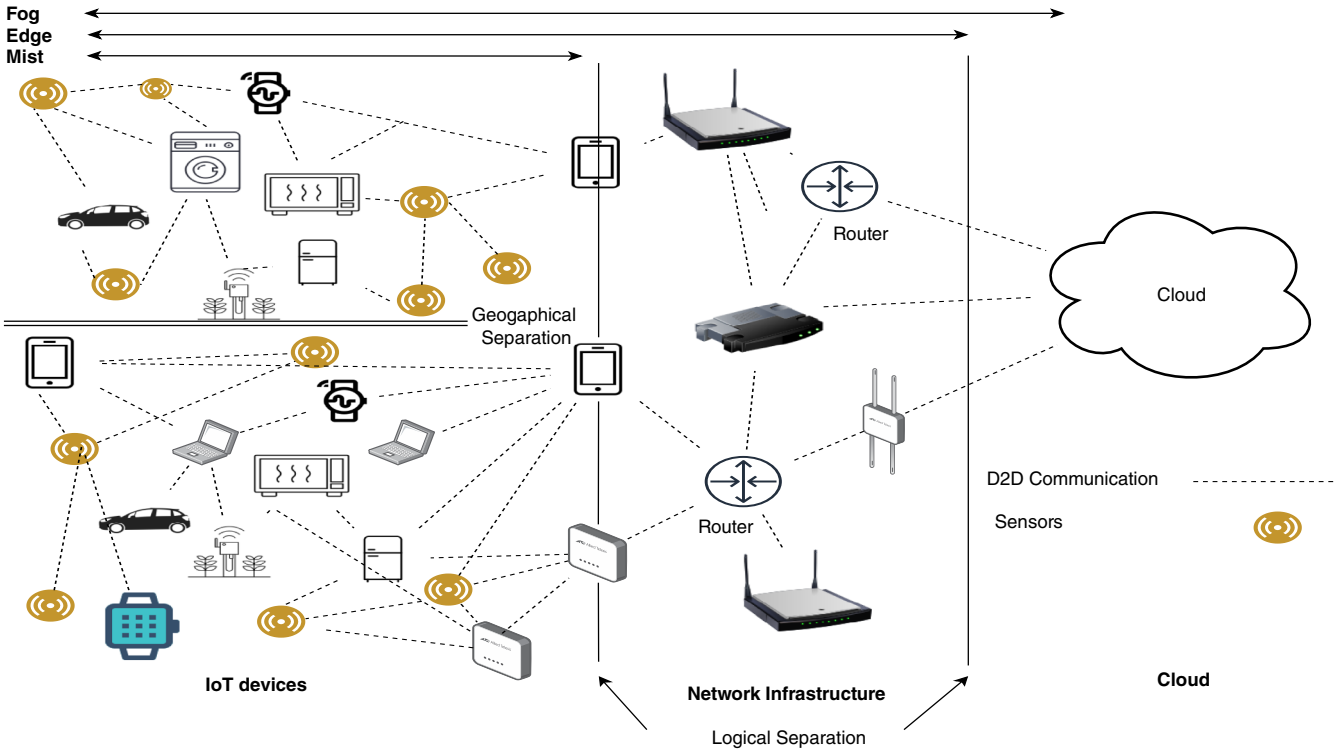


FIGURE 1 Mist networks

use more persistent nodes of the network infrastructure. Maintaining and organizing thousands of nodes in a distributed network is a complicated task, but it can enable the service migration and task execution to Mist nodes and the usage of a lot of free resources in a distributed and efficient way. Today, it seems that the most crucial obstacle to Mist development is network management, and the lack of efficient ways to discover, organize, and share distributed resources in order to allocate them to different tasks.[17]

Mist computing is generally defined as distributed processing and parallel computation in IoT nodes to reduce latency and improve the efficiency of the network in case of intermittent Internet connectivity. In order to run services and execute tasks, distributed resources of nodes should be allocated to the tasks. The first steps of the resource allocation are discovering and organizing the available resources as a mIoT network, which can have thousands of heterogeneous nodes. Many techniques have been proposed to discover, share, and allocate resources in IoT networks, but none of them are defined for highly heterogeneous massive-scale Mist networks. Because Mist networks can include thousands of nodes, revealing the resource-rich nodes in each geographical area that are available to help other nodes is a complicated task due to the heterogeneity and the dynamicity of the Mist networks. Mist networks are highly dynamic, and nodes can join, disjoin, and move at any time, which makes it difficult to allocate a node to execute a long-running task or host a service for a long time. In addition, because each node can use different network connections and services, the network is more complicated than all other Edge computing paradigms. Besides, the available resources of each node can vary in time, and nodes can be enabled or disabled temporarily. Figure 1 shows different layers of IoT networks and the place of Mist nodes.

Generally, IoTs are highly heterogeneous networks; the different nodes are equipped with the resources they need to perform their specific tasks.[20] An example is a network at a university, where devices like laptops, smartphones, smartwatches, and environmental sensors can use the same network infrastructure to connect to both the Internet and other devices. The network is highly heterogeneous, not only due to different physical resources but also the applications running on the nodes and the services required by the nodes.[21] To recognize the resource-rich nodes that can offer their resources in such a diverse and massive-scale network, the nodes were compared locally in order to identify the appropriate candidates. Although the most important parameters are the available physical resources like energy, computation power, and network interfaces, there are also other parameters that can affect which nodes are most suitable. As IoT networks are typically categorized as an ultra-dense network (UDN),[22] the requirements of the physical layer also need

to be considered. In addition, because of the different applications and services that are demanded by each node in a shared network infrastructure, the application layer requirements need to be considered. In order to identify the nodes that are eligible for executing specific tasks or hosting specific services, various methods are available in the literature. However, this article propose a method to assess and identify nodes that are, in general, good candidates for providing their resources to share with others when compared with their neighbors. The selected nodes are relatively more suitable than their neighbors to execute tasks, not only based on their resources but also to satisfy application and physical layers requirements.

The primary objective of this article was to design a node assessment method for highly heterogeneous mIoT-Mist networks. The method proposed in this article contributes to a larger goal of designing a hierarchical clustering method that can be used to build an overlay network for sharing resources in IoT networks. The proposed method can find the most resource-rich nodes, which are also most able, among their neighbors, to support the physical and application layer requirements. The proposed method uses eight comparative parameters among neighbors to select the most eligible nodes. These parameters are then combined to calculate a score for each node. The nodes explore their neighbors to select the most eligible neighbor, and then the best nodes are selected based on votes from neighbors. The results show that the proposed method can identify the nodes which are more powerful and relatively are in a better situation when compared with neighbors in their geographical area. By only considering information obtained from one-hop neighbors in the choice of suitable nodes, the overhead is kept at a minimum and scales well to large networks, unlike methods that need information about a whole network. The proposed method also considers the requirements of the application layer, which can help to improve the efficiency of service placement, service migration, and data fusion. Because some parameters are more important than others, a point-wise learning-to-rank method was used to model the relationship between these eight parameters and heuristically predict a final suitability score. In addition, a second chance method was proposed to reorder the neighbors in order to balance the load among the most suitable nodes.

The main contributions of this article are summarized below:

- The provided model is used to identify resource-rich nodes which have a better situation than their neighbors to participate in Mist computing in highly heterogeneous mIoT-Mist networks with hundreds or thousands of nodes.
- Combining parameters that can inform decisions needed to increase the efficiency of a highly heterogeneous IoT network based on different network layer requirements.
- Designing an assessment method that can determine the suitability of nodes in IoT-Mist networks to offer their resources to help their local neighbors.
- Designing a second chance method to avoid selecting only the most powerful nodes and give a chance to the nodes which are relatively good compared with the best nodes.

The rest of the article is structured as follows. In Section 2, related works are reviewed to show the importance of the proposed method. In Section 3. In Section 4, the proposed method is evaluated to show its efficiency. Finally, Section 5 proposes some future works and concludes the article.

2 | RELATED WORK

Resource discovery is a big challenge in many distributed systems[23] and P2P networks.[24] There are several survey articles on resource sharing,[25] resource discovery mechanisms, frameworks,[26,27] and resource management, specifically in new computing paradigms like Edge and Fog computing.[28] In many applications, storage and contents are shared as a resource with other nodes. In Reference 29, Wang introduces SoFA, which is a network resource management model for peer-to-peer semantic communities to organize contents and storage as resources. The challenges that are solved include a method to find appropriate resources effectively and quickly, obtaining and maintaining a huge number of shared good resources, and finally designing a routing mechanism to access the shared resources effectively in P2P networks. They focus on two first problems and the introduced method is used to recognize the location and quality of the resources to solve them, respectively. In SoFA, an autonomous peer model collects information and communication and evaluates them with the help of other local APs to build an expert-driven autonomic semantic community to evaluate the trust based on semantic similarity, history, and time effect.

Discovering resources through distributed systems is considered as a key service in such systems to dedicate and share resources. In Reference 30, the authors use a hybrid resource discovery approach for P2P Grid networks in which they integrate spanning trees for information propagation and epidemic algorithms. They use a mathematical model to predict the process of information dissemination and a model to evaluate the quality of the prediction. They also show that the proposed method is failure resistant in scenarios in which up to 50% of peers are failed in a short time. The authors of Reference 31 study the performance of classic flooding, random walk, and gossip-based resource discovery algorithms in mobile P2P networks, and improved the algorithms' performance in mobile ad hoc networks (MANET) networks. They also propose to improve the algorithms to work in MANET. They compare the algorithms on success rate, energy consumption, response time and overhead and QoS metrics by using an NS-2 Simulator. Talpur et al,[32] introduce an IoT network infrastructure that shares services of IoT nodes as resources to reduce monetary costs. They share the services of some sensors among multiple users and introduce some methods to avoid data loss and spoofing. They simulated their architecture by NS3 and added ubiquitous homomorphic encryption to validate nodes, users, and data. They applied the proposed model to use shared-nodes validated with a geographic saturation model and they also tried to reduce the probability of successful cryptanalytical attacks.

In Reference 33, Abedin et al designed a model for Edge and Fog networks, which is used for node pairing to address utility pairing and matching problems based on Irving's matching algorithm. They use the algorithm to ensure stable IoT node pairing. They provide a model for node pairing in IoT-Fog enabled networks based on modeling the problem as a "one sided stable matching game" and they define a utility-based preference list to pair IoT nodes. Azam et al[34] present a model that is used for resource prediction, resource estimation, reservation, and pricing for IoT customers based on their characteristics. Their model is used to predict and estimate the resources that each customer needs in the Fog, in addition to locating, reserving, and estimate the cost of using the resources from a pricing model. They use CloudSim toolkit and Java to evaluate their proposed model. In Reference 35, the authors present a status-aware and stability-aware mobile device selection method which is used to find optimal mobile devices in edge networks. By storing the status and historical characteristics of mobile devices and using a cloud model to evaluate the stability of each device, the optimal device will be selected to help other neighbors by hosting services and executing tasks.

Fernando et al[36] introduced a work-sharing model called "HoneyBee" to balance the load among heterogeneous mobile nodes based on a well-known work-stealing method. They focus on short-term available mobile nodes that are joining and leaving based on proactive worker and opportunistic delegator concepts. They consider heterogeneity of nodes, unknown capability and dynamism as challenges to address and present a model which use an adaptation of the well-known work stealing method to allocate tasks to heterogeneous mobiles nodes considering dynamicity. Short-term goals and using the advantages of resource sharing on arising the new available resources are two main factors to introduce the model. They report up to 71% optimizing energy consumption in the network. In Reference 37, the authors introduce a search engine in IoT networks which helps to discover objects that are able to store the data from sensors as part of their proposed method. The search engine has three layers, including run-time monitoring the equipment of devices, distributively storing the data, and providing access by IoT devices to data.

Although there is a paucity of literature on the problem of resource discovery in highly heterogeneous Mist networks, the available articles are briefly reviewed. Vasconcelos et al[10] introduce a model to use the computational resources in three different layers of the Cloud, Fog, and Mist. They evaluate the cost, bandwidth and latency of each layer based on available resources and topology of the network to determine the best layer to execute tasks. They also introduce a method that reduces the time of exploring eligible nodes among client devices and their neighbors. The authors in Reference 38 also review how self-awareness can help Fog and Mist networks as a cyber-physical system. They mention that there are two central aspects of self-awareness; namely, attention and situation-awareness. They explain that monitoring the performance of the system to recognize changes is an important aspect. In addition, attention can help to balance the tasks which need resources in resource-constrained CPSs. Barik et al[39] introduce MistGIS, which is used to process geographical information system (GIS) data in Mist and Edge and store the data in the cloud. They use a K-means model to cluster nodes in different geographical areas and evaluate how many Fog and Mist nodes are needed based on their resources to analyze various GIS datasets to reduce the overhead of the cloud.

3 | THE PROPOSED METHOD

IoT is a network infrastructure that connects billions of CPSs, such as sensors, cellphones, home appliances, cars, and homes, to each other in order to gather information and perform tasks. Computation power of IoT networks

can be provided by different machines remotely or locally. When compared with traditional cloud resources, such as remote computation resources, the Mist can provide more accessible and affordable resources with greater efficiency by performing computation closer to the user equipment. As IoT-Mist networks contain many highly heterogeneous nodes, recognizing the most resource-rich nodes is a distributed task, and is needed in order to place tasks and services on the most appropriate nodes. In addition, because the IoT nodes are heterogeneous in several aspects, only considering a single feature, like available physical resources, may result in a suboptimal placement of services and computation tasks. Therefore, several requirements need to be considered when selecting the most suitable nodes, and the overhead must be minimal with respect to the size of a massive-scale network, thereby eliminating the possibility of using a centralized task. The proposed method keeps the overhead to a minimum by executing the assessment method as a distributed task in which each node is responsible for comparing itself with its neighbors and selecting the most suitable node. It is demonstrated in this article that the proposed method generally finds the resource-rich nodes in each geographical area, and succeeds in identifying the most suitable nodes in a mIoT network with thousands of nodes. There are, however, some assumptions which need to be considered when using the proposed method:

- The nodes are aware of their own energy and hardware resources, including processing power, network interfaces, RAM, and storage. In addition, each node is aware of which services it requires, and whether the node is a data generator or a data consumer of those services. In this article, an application A of Node_i, specifies that Node_i connects to a specific service related to Application A to send and receive data. In Section 3.2.1, applications are categorized based on their resource consumption. The nodes know which of these categories their applications belong to compare their applications with each other. The user can specify the category of applications and amount of resources that they are expected to consume or they can be specified based feedback systems, resource estimation models,[40] or resource provisioning models.[41]
- The nodes are heterogeneous in several aspects, including energy, CPU, RAM, Storage, network interfaces, applications, and the services that they use. Each node is aware of its own resources, applications and the resources and applications of its one-hop neighbors by broadcasting or via piggyback methods. Although the proposed node assessment method is executed in the setup phase, events like mobility, node failure, significant changes in the available resources of a node, or QoS problems can trigger the nodes to execute the assessment model again locally by asking neighbors to broadcast their current state. As the nodes are only assessed locally, there is no need to trigger the reassessment globally.
- The resource metrics are either using standard units among the nodes or are convertible. As the proposed method is a comparative model, having the same units for the parameters for each node is essential, but nodes can have different underlying providers of the raw values and convert the information before broadcasting to their neighbors.

Eight parameters are defined such that they help identify the most suitable nodes, in a geographical area, based on requirements involving several layers in the network stack. These can then be used to place tasks or services or manage other nodes such as cluster heads in clustering techniques. These nodes can also be used for caching and data fusion, which can be executed in Mist networks to reduce overhead of the network. In the definitions, N_i is the node that wants to calculate its score, with being $N_j, \dots, N_z$ the array of N_i neighbors. In addition, C is used as the number of its neighbors. In all parameter definitions, N_n is the neighbor of N_i, if they have at least one common network interface, like Bluetooth or Zigbee, and their distance is lower than the radio range of the network interface, except for WiFi neighbors who need to communicate with the same access point to connect. As different parameters can have some impact on the requirements of all layers, it is hard to allocate each parameter to a layer, but categorize the parameters based on their highest impact in this section. For calculating the score and selecting the node that is the most capable of offering its resources, the method uses a process:

- **Broadcast parameters**: First, the nodes broadcast their parameter values, like energy, physical resources, coverage area, and its applications to all neighbors.
- **Calculate and broadcast score**: The nodes then use the proposed method to combine the received parameter values into a score signifying its suitability for being selected. The nodes then broadcast their score to their neighbors.

- **Rank and select neighbor based on score**: The nodes then sort their neighbors and themselves according to the scores they receive and select the top ranking node.
- **Inform the selected node**: Having selected a neighbor, the node informs that neighbor that it has been selected.

3.1 | Parameters to support network layer requirements

The goal of the network layer parameters is to improve the efficiency of the network connections among nodes when considering aspects like QoS, and network lifetime. In this section, the parameters of the proposed method are described, these are all designed to improve the efficiency of network layer services.

3.1.1 | Energy

In IoT-Mist networks, most of the nodes are resource-constrained, especially in terms of their energy resources for battery equipped devices. While some nodes are connected to infinite energy resources like a power grid, the method generally consider IoT-Mist nodes to have a limited energy supply, which should be consumed efficiently in order to prolong the life of the device. Each network has a threshold between first-node-die and last-node-die, which is used to evaluate the efficiency of the network, so prolonging the life of each node will also help to prolong the life of the network. In addition, node failure caused by exhausted energy supply can result in data loss, which is a critical problem for reliability-sensitive applications like those found in healthcare. Therefore, energy resource is one of the most crucial parameters to consider when assessing a node. The method, therefore, ensure that nodes with a higher energy surplus than other neighbors will have a higher chance of being selected. This is important because the selected nodes will be under an additional load from processing and data forwarding when helping their neighbors, causing them to consume more energy and reducing the lifetime of nodes if they have a limited energy supply. $\text{Energy}_{\text{rest}}(N_i)$ is the remaining energy of N_i, which in each trigger is broadcast by the node to be compared in neighbors. When comparing the energy of N_i with the energy of its neighbors by Equation (1).

$$\text{Energy}_{\text{cmp}}(N_i) = \frac{\text{Energy}_{\text{rest}}(N_i)}{\left(\sum_{n=j}^{z} \text{Energy}_{\text{rest}} N_n\right) + \text{Energy}_{\text{rest}}(N_i)}. \tag{1}$$

The result is $0 \leq \text{Energy}_{\text{cmp}}(N_i) \leq 1$ which is the suitability of the nodes among its neighbors in terms of the energy resource. When $\text{Energy}_{\text{cmp}}(N_i)$ is approaching 1, it means that the node has the best energy resource among its neighbors.

3.1.2 | Coverage

IoT networks are becoming more quasi-ad hoc networks, as most of the network interfaces are enabled to use device-to-device (D2D) communication with their neighbors. As explained in Reference 42, D2D is one of the main communication technologies in the IoT ecosystem that devices will use to communicate with each other, autonomously, without the need to use a centralized system. Today, many of the most common communication technologies, like WiFi Direct, Bluetooth, and Zigbee are mainly designed to support D2D communication. In addition, new technologies are increasingly supporting D2D as one of the essential methods to improve the efficiency of the networks. An example is 5G, which is one of the most important IoT cellular communication technologies, and has been designed with D2D in mind.[43] Besides, current cellular network communication technologies, such as 4G and LTE are D2D-enabled by FlashLinQ.[44] As the number of IoT nodes is growing, maintaining the efficiency of the network-infrastructure-dependent networks will get harder, due to network overhead and high resource consumption. Network-infrastructure-independent networks, like D2D communication technologies, can provide a more flexible and efficient network infrastructure that mimics ad hoc networks. In quasi-ad hoc networks, the coverage of the nodes is an important parameter to improve connectivity, as nodes can connect to more devices directly via D2D technologies. Having more connections means more routes, more options to apply different priorities to support QoS metrics like reliability and reducing delay, and more solutions

to support the dynamism and mobility of the network. The radio range of each node specifies the coverage area, but in several IoT nodes there is more than one type of network interface that communicate with neighbors. As an example, a smart cellphone may be equipped with WiFi, WiFi direct, Cellular D2D, and Bluetooth. The nodes can use different types of network interfaces as both the receiver and sender, which means that the maximum radio range among network interfaces can be considered as the general coverage area of the node. To compare the coverage of each node with its neighbors use Equation (2).

$$\text{Coverage}_{\text{cmp}}(N_i) = \frac{\text{Coverage}_{N_i}(\max(\text{Interfaces}(N_i)))}{\left(\sum_{n=j}^{z} \text{Coverage}_{N_n}(\max(\text{Interfaces}(N_n)))\right) + \text{Coverage}_{N_i}(\max(\text{Interfaces}_{N_i}))}. \quad (2)$$

As Interfaces(N) returns the radio ranges of all interfaces of node N, $\text{Coverage}_{\text{cmp}}(N)$ is a value between 0 and 1 which shows the situation of the node coverage compare to its neighbors

3.2 | Parameters to support application layer requirements

IoT is generally a service-based network,[45] which means that services in one or more machines are responsible for serving end-user devices to execute their tasks and process their data, and end users that need a service need to communicate with the host of that service. Traditionally, IoT services run on the cloud, but because the cloud is geographically far from end-user nodes, this can cause inefficiencies. A solution is to migrate the services to machines that are closer to the end users or even resource-rich end users as in Mist networks. Most IoT services need significant resources in order to be executed, which means that the nodes with more resources are more appropriate as the host of a service. In addition, how close the host of a service is to end users is important to reduce delays and improve resource consumption. Therefore, the selected nodes should compare favorably, in both proximity, and resources when considering the application requirements against their neighbors.

3.2.1 | Hardware resources

Physical resources (generally known as hardware resources) are one of the most important parameters to identify suitable nodes. For this method, each node has a measurable amount of processing power, storage, and forwarding rate, which indicates the amount of resources available to execute tasks. In this method, the nodes will only broadcast the available resources they can share. Three types of hardware resources are considered:

Forwarding rate is the number of bits per second that can be transmitted through the node. A node can have different forwarding rates for different network interfaces. As each node in quasi-ad hoc networks can be used as a relay, the forwarding rate is an important factor. While there is another parameter called "interfaces," which compares the availability of network interfaces by type, it is worth mentioning that a single type of network interface can have different forwarding rates based on various protocols, standards, physical obstacles, or interference, so each node needs to consider its forwarding rate separately.

Processing power: All tasks require some CPU cycles to be executed, some more than others. In order to run services on a node, considerable computation power might be required. As IoT nodes are generally resource-constrained, discovering nodes with an abundance of computation power can identify them as able to run both light-weight tasks like data fusion and heavy-weight tasks like containers. RAM is another parameter to consider, as some tasks require significant amounts of RAM resources. In this article, both CPU and RAM are referred to as the processing power parameter.

Storage: Especially in Big data IoT applications, nodes and services might need to store huge volumes of data to save, process, and forward. Having more storage in some applications helps to gather more information and process them before transferring, which would reduce the network overhead. In addition, there may be cases where applications need to store data in order to trigger services like

database containers. Storage capacity is, therefore, a useful parameter to determine if a node is an eligible candidate in case of storage-consuming applications.

To compare the hardware resources of each node with its neighbors Equations (3) to (6).

$$\text{Forwarding}_{\text{cmp}}(N_i) = \frac{\text{Forwarding}(N_i)}{\left(\sum_{n=j}^{z} \text{Forwarding}(N_n)\right) + \text{Forwarding}(N_i)}, \tag{3}$$

$$\text{Computation}_{\text{cmp}}(N_i) = \frac{\text{Computation}(N_i)}{\left(\sum_{n=j}^{z} \text{Computation}(N_n)\right) + \text{Computation}(N_i)}, \tag{4}$$

$$\text{RAM}_{\text{cmp}}(N_i) = \frac{\text{RAM}(N_i)}{\left(\sum_{n=j}^{z} \text{RAM}(N_n)\right) + \text{RAM}(N_i)}, \tag{5}$$

$$\text{Storage}_{\text{cmp}}(N_i) = \frac{\text{Storage}(N_i)}{\left(\sum_{n=j}^{z} \text{Storage}(N_n)\right) + \text{Storage}(N_i)}. \tag{6}$$

To integrate RAM and CPU as two parameters which affects computation power, Equation (7) are used, which indicate the processing power of the node compare to its neighbors.

$$\text{Processing}_{\text{cmp}}(N_i) = \frac{\text{Computation}_{\text{cmp}}(N_i) + \text{RAM}_{\text{cmp}}(N_i)}{2}. \tag{7}$$

Although the available resources parameter can be used to show the eligibility of a node, in reality, if nodes want to offer their resources, they need to evaluate the resource requirements of their neighbors. Each type of application needs different resources, which can be determined by the user or can be estimated based on various resource estimation methods.

Table 1 is used to classify different types of applications as "application" defined in Section 3. Each application is classified into one of eight groups based on its requirements, as listed in Table 1. In this table, "*Yes*" means that the application needs a significant amount of the specific resource to run in the node, and "*No*" means the application does not need a significant amount of the resource in the node. In the equations, "*Yes*" is translated to 1 and "*No*" is translated to 0. The total number of possible applications per node is A, and each node can have up to 1 application per type. If $A = 8$, then a binary array shows each node has what types of applications. As an example, 1101001 shows that N_i has Application types of 1, 2, 4, and 8. If N_i has C number of neighbors, first, the number of requirements of all neighbor applications is counted separately, and designated $\text{Forwarding}_{\text{Nei-apps}}(N_i)$, $\text{Processing}_{\text{Nei-apps}}(N_i)$, and $\text{Storage}_{\text{Nei-apps}}(N_i)$. These are based on Equations (8) to (10), respectively. As $N_n(\text{App}_p)$ returns 1 if the application of p of Node n needs the resource significantly and returns 0 if it does not need the resource.

$$\text{Forwarding}_{\text{Nei-apps}}(N_i) = \sum_{n=i}^{z} \sum_{p=1}^{8} \text{Forwarding}_{\text{rate}}(N_n(\text{App}_p)), \tag{8}$$

$$\text{Processing}_{\text{Nei-apps}}(N_i) = \sum_{n=i}^{z} \sum_{p=1}^{8} \text{Processing}_{\text{rate}}(N_n(\text{App}_p)). \tag{9}$$

$$\text{Storage}_{\text{Nei-apps}}(N_i) = \sum_{n=i}^{z} \sum_{p=1}^{8} \text{Storage}_{\text{rate}}(N_n(\text{App}_p)). \tag{10}$$

TABLE 1 Different types of applications

Applications	Resource Requirements			Example of the Type of Application in Real World
	Forwarding	Processing	Storage	
Type 1	No	No	No	Environmental sensors like temperature or motion sensors
Type 2	No	No	Yes	Big data gathering like crowd-sensing
Type 3	No	Yes	No	Healthcare data processing
Type 4	No	Yes	Yes	Image processing
Type 5	Yes	No	No	Environmental high rate sensors like Camera
Type 6	Yes	No	Yes	Big data fusion applications
Type 7	Yes	Yes	No	Pattern recognition methods
Type 8	Yes	Yes	Yes	Multimedia processing

Total resource requirements of neighbors is calculated by Equation (11).

$$\text{Sum}_{\text{app}}(N_i) = \text{Forwarding}_{\text{Nei-apps}}(N_i) + \text{Processing}_{\text{Nei-apps}}(N_i) + \text{Storage}_{\text{Nei-apps}}(N_i). \tag{11}$$

To show the weight of each resource compare to others to select the eligible nodes, use Equations (12) to (14).

$$\text{Forwarding}_{\text{Weight}}(N_i) = \text{Forwarding}_{\text{Nei-apps}}(N_i)/\text{Sum}_{app}(N_i). \tag{12}$$

$$\text{Processing}_{\text{Weight}}(N_i) = \text{Processing}_{\text{Nei-apps}}(N_i)/\text{Sum}_{app}(N_i). \tag{13}$$

$$\text{Storage}_{\text{Weight}}(N_i) = \text{Storage}_{\text{Nei-apps}}(N_i)/\text{Sum}_{app}(N_i). \tag{14}$$

Finally, to compare the available resources in node N_i with requirements of applications of its neighbors, Equation (15) is used, in which $0 \leq \text{Resource}_{\text{cmp}}(N) \leq 1$. $\text{Resource}_{\text{cmp}}(N)$ show that how much the node is eligible compare to its neighbors in term of hardware resources to support requirements of its neighbors.

$$\begin{aligned}\text{Resource}_{\text{cmp}}(N_i) = &(\text{Forwarding}_{\text{Weight}}(N_i) * \text{Forwarding}_{\text{cmp}}(N_i)) \\ &+ \text{Processing}_{\text{Weight}}(N_i) * \text{Processing}_{\text{cmp}}(N_i)) + \text{Storage}_{\text{Weight}}(N_i) * \text{Storage}_{\text{cmp}}(N_i)).\end{aligned} \tag{15}$$

3.2.2 | Application similarity

Although it is hard to predict the similarity of applications, such as two neighbors watching the same movie in order to use caching methods, the similarity of application types can increase the chance to use methods like caching, data fusion, or hosting a relevant service close its users. As an example, if node N_i is selected by 20 nodes, all of which are either a generator or consumer of the same type of application, like temperature sensing, N_i has a good chance of reducing the resource consumption and improving the efficiency of the nodes by hosting a service which fuses data before sending to the cloud. In the proposed method, a weighting model is used to give priority to specific applications, which can be helpful in supporting QoS by giving more priority to more important applications, such as real-time applications. If an application has a higher priority than others, a node that is more similar to neighbors for this application type has a higher chance of being selected. In the method, the total amount of weights should be 100. For example, the weight of all applications can be 12.5, which indicates that the priorities of all applications are equal. In Equation (16), $\text{App}_b(\text{Neighbors}(N))$ calculates the total number of App b for neighbors of node N.

$$\text{App}_b(\text{Neighbors}(N_i)) = \sum_{n=j}^{z} \text{App}_b(N_n). \tag{16}$$

Two different situations are available to compare the similarity of applications of neighbors formulated in Equation (17).

- If $\text{App}_b(N_i) = 1$: The score increases up to maximum as the number of neighbors with increase
- If $\text{App}_b(N_i) = 0$: The score increases from the minimum as the number of neighbors have App_b

Mathematically, two conditions are described in Equation (17).

$$\text{Sum}_{\text{Neighbors}(N_i)}\text{App}_b(N_i) = \text{Weight}(\text{App}_b)\left(\frac{\text{App}_b(\text{Neighbors}(N_i))}{C}\right)\sum_{n=j}^{z}\begin{cases}\text{App}_b(\text{Neighbors}(N_i)) & \text{if } \text{App}_b(N_i) = 1\\ C - \text{App}_b(\text{Neighbors}(N_i)) & \text{otherwise.}\end{cases}. \tag{17}$$

Equation (17) shows that when N_i calculate its score, it if has Application b then it increase score, otherwise increasing or decreasing the score is related to the number of neighbors which have the Application b. If there are a lot of neighbors which have the Application b then the score is increased, otherwise the score is decreased.

$$\text{App-Similarity}_{\text{cmp}}(N_i) = \frac{(\text{Sum}_{\text{Neighbors}(N_i)}\text{App}_b(N_i))/(A \cdot C)}{100/C}. \tag{18}$$

To show the eligibility of the node in case of similar types of applications with its neighbors, Equation (18) result between 0 and 1.

3.2.3 | Neighbors

The number of neighbors can affect the performance of the nodes, as more physical resources are needed to host services and execute tasks to help neighbors. In addition, a node in a crowded area will have a higher chance of having more neighbors at some points during its lifetime, because of the dynamicity of the network and node movement. In order to compare the number of neighbors, the proposed method uses Equation (19), which gives a value between 0 and 1. In order to ensure that this parameter has the same range as the other parameters, the result is subtracted from 1. If $\text{Neighbor}_{\text{cmp}}(N_i)$ is approaching 1, it means that the node is more suitable than other neighbors. Comparing the available resources and the number of users is done by a machine learning method, as explained in Section 3.5.

$$\text{Neighbor}_{\text{cmp}}(N_i) = 1 - \frac{\text{Neighbors}(N_t)}{\left(\frac{\sum_{n=j}^{z}\text{Neighbors}(N_n)}{c}\right) + \text{Neighbors}(N_i)}. \tag{19}$$

3.3 | Parameters to support physical layer requirements

IoT networks are generally considered as UDNs in many IoT applications, such as crowdsensing and smart cities.[22] In such applications, the physical layer requirements should be supported to reduce the interference. Three parameters are considered when selecting nodes that can improve satisfying physical layer requirements listed below:

3.3.1 | Interfaces

An IoT node can have multiple network interfaces which can determine its ability to communicate with other nodes and its reliability to transfer data by using different network interfaces in case of failures and inefficiencies. As the host of services and a place to execute tasks, the nodes with multiple interfaces can connect to more nodes and reduce interference by changing the active network interface. In the comparison, Bluetooth, Zigbee, Wi-Fi, Wi-Fi Direct, and Cellular D2D

are considered as possible network interfaces for each node. Having more interfaces gives a node the ability to discover more nodes, support them, or use their resources efficiently. In addition, with more interfaces it is possible to find better and more efficient routes in a heterogeneous network. Furthermore, connectivity impairment can be solved by switching between different network connections. In the proposed method, the five types of interfaces are expressed by a Boolean array for each node. The number of interface types is equal to $\text{Interface}_{\text{type}} = 5$. In order to compare the number of interfaces with neighbors, the proposed method uses Equation (20). $\text{Interface}_a(N)$ is equal to 1 if node N has the network interface of a.

$$\text{Interface}(N_i) = \sum_{a=1}^{\text{Interface}_{\text{types}}} \text{Interface}_a(N_i)/\text{Interface}_{\text{types}}. \tag{20}$$

To sum up the whole network interfaces of the neighbors of N_i, Equation (21) is used.

$$\text{Interface}_{\text{Neighbors}}(N_i) = \sum_{n=j}^{z} \sum_{a=1}^{\text{Interface}_{\text{types}}} \left(\begin{cases} 1 & \text{if } \text{Interface}_a(N_n) = \text{Interface}_a(N_i) = 1 \\ 0 & \text{otherwise} \end{cases} \right). \tag{21}$$

To calculate the similarity of network interfaces among neighbors, the following equation is used:

$$\text{Interface-Avg}_{\text{Neighbors}}(N_i) = \frac{\text{Interface}_{\text{Neighbors}}(N_i)}{c \cdot \text{Interface}_{\text{types}}}. \tag{22}$$

To calculate the similarity of network interfaces between N_i and its neighbors, Equation (23) is used:

$$\text{interface}_{\text{cmp}} = \frac{\text{Interface}_{\text{Neighbors}}(N_i)}{\text{Interface-Avg}_{\text{Neighbors}}(N_i) + \text{Interface}_{\text{Neighbors}}(N_i)}. \tag{23}$$

The result of Equation (23) is between 0 and 1 and show the similarity of network interfaces between N_i and its neighbors.

3.3.2 | Proximity

Interference is an important problem in UDN IoT networks, and with the multiple interfaces in a node, the problem is even more critical. The selection should, therefore, consider the eligible nodes as bottlenecks if several nodes connected to them. As a fundamental solution to reduce interference, each node can reduce the radio range of each network interface by reducing the power used to communicate with other nodes. In addition, reducing radio range helps to decrease energy consumption simultaneously, as the wireless transmission is one of the leading causes of energy drain in IoT networks. To support this solution, nodes that are closer to their neighbors should be selected to be able to reduce radio range. The energy consumption of a wireless transmission has a high correlation with the distance and the communication technologies used, following Equation (24).[46] Here, $d(N_i, N_j)$ is the distance between nodes N_i and N_j. Equation (25) is used to calculate the average proximity of N_i and its neighbors in a 2D environment. Equation (26) is used to compare the proximity of the node with its neighbors, which is between 0 and 1.

$$E_t = F(d(N_i, N_j)^2). \tag{24}$$

$$\text{Proximity}(N_i) = \frac{\sum_{n=j}^{z} \sqrt{(\text{Pos}_x(N_i) - \text{Pos}_x(N_n))^2 + (\text{Pos}_y(N_i) - \text{Pos}_y(N_n)}}{C}. \tag{25}$$

$$\text{Proximity}_{\text{cmp}} N_i = 1 - \frac{\text{Proximity}(N_i)}{\left(\sum_{n=j}^{z} \text{Proximity}(N_n)/C \right) + \text{Proximity}(N_i)}. \tag{26}$$

3.3.3 | Density

One of the problems of quasi-ad hoc networks is blind spaces if the network is highly dynamic, this could even be a critical problem. To improve the connectivity, nodes that can connect more nodes should have a higher chance of being selected. Density measures, proportionally, how many nodes are available in the area covered by a node. In order to achieve the most extensive coverage of an environment, a node that can support more nodes based on its coverage should have a higher chance of being selected. Equation (27) is used to calculate the density of the area which is covered by N_i. To compare the density of the N_i with neighbor's density, the proposed method uses Equation (28).

$$\text{Density}(N_i) = \frac{\pi(\text{Coverage}(N_i))^2}{C}. \tag{27}$$

$$\text{Density}_{\text{cmp}}(N_i) = \frac{\text{Density}(N_i)}{\sum_{n=j}^{z} \text{Density}(N_n) + \text{Density}(N_i)}. \tag{28}$$

3.4 | Correlations among parameters

While eight parameters are considered when selecting the optimal nodes with respect to layer requirements, not all parameters are equally important in all situations. Therefore, these parameters must be prioritized when comparing the nodes. A simple solution is to use a weighting model or optimization methods to give different priority to each of the parameters, but in some cases, there is a trade-off among two or more parameters based on their values. This trade-off makes the problem more complicated, as the weights of parameters should be recalculated in each case. Therefore, instead of using a weighting model, the priority order list shown in Table 2 is used. In Table 2, the principal correlation and priorities among parameters, based on human understanding of the problem, are given. In addition, the table reports how much effect each parameter has on supporting each layer's requirements, as some parameters can affect multiple layers. The priority order list is the foundation stone of an initial dataset for the learning-to-rank method provided by a human expert, which is a method described in Section 3.5. To show the whole idea and parameter correlations, a dataset was built by the authors by answering questions based on the table, which is available in Reference 47 as a CSV-formatted file named "Machine Learning Dataset.csv." In order to convert the idea from numeric to human-understandable format, a semi-fuzzy method is used to express the idea in a dataset. Each input and output parameter can have three values: Low, Medium, and High. Based on these parameters, all input values are between 0 and 1. By having eight parameters and three values for each, there are 6561 possible statements. A random method was used to select only 500 statements, and they have been answered by an expert to show the correlation based on Table 2 and the priorities. In addition shown in Table 2 are the effects of each parameter on energy consumption and QoS. In addition to presenting the priority of each parameter, the rules are only formed by logical conjunctions. However, the score as the output of the method used in this article is produced by the machine learning model explained in Section 3.5.

Node N_i calculates its score and broadcasts it to the neighbors $N_j \ldots N_Z$ via several different network interfaces in the setup phase. In addition, N_i uses the scores received from its neighbors, in addition to its score, to create a sorted list of scores. The first node of the list is selected as the most eligible neighbor, and it is informed by the node. The machine learning method learns a general model that combines information from all parameters. Because of the way the dataset used to train the machine learning model is built, the model will incorporate the priority of the parameters. Here are the eight rules used to describe the trade-offs between the parameters that are considered when building the dataset used to learn the machine learning model:

1. Energy and physical resources have the highest priority. When they have different values, Energy is more important.
2. There is a trade-off between Neighbors and Energy; when Energy is high, having more neighbors increases the score; if Energy is low then fewer neighbors can increase the score.
3. The trade-off between Neighbors and Physical resources shows that if Neighbors is high, Physical resources should be so high, and if Neighbors is low, then the impact of physical resources is lower on decreasing or increasing the score.

TABLE 2 Correlations of parameters, priority, and impact

		Impact on Network Parameters		Impacts on Layers Requirements		
Parameter	**Priority**	**Resource Consumption**	**Quality of Service**	**Application**	**Network**	**Physical**
Energy	1	High	Low	Low	High	Medium
Physical resources	1	High	Medium	High	High	Medium
App-similarity	2	High	Medium	High	Low	Low
Neighbors	3	High	Medium	Medium	High	High
Interfaces	4	Medium	High	Medium	High	High
Coverage	5	Medium	Low	Low	Medium	High
Proximity	6	Medium	Low	Low	Medium	High
Density	6	Medium	Low	Low	Low	High

4. When the Application similarity is high between N and its neighbors, then the physical resources should also be high to increase the score. When it is low, higher physical resources can increase the score further.
5. There is a trade-off between neighbors, rule 3 and rule 4, as when the number of neighbors is high, then rule 4 should be lower to increase the score.
6. There is a trade-off between Interfaces and Coverage, as when Coverage is high, higher Interfaces can increase the score more than the situation in which the Coverage is lower in the same value of Interfaces.
7. There is a trade-off between Interfaces and Neighbors, as when the number of neighbors is high, higher interfaces increase the score further.
8. There is a trade-off between Proximity and Density, as they have an equal priority in the same values, but Proximity has a higher priority when values are different.

Based on Table 2 and the rules, an expert answered a learning model that is available in Reference 47. Then the model is used by a learning-to-rank method explained in Section 3.5.

3.5 | Learning-to-rank method

In this article, a ranking method is used to select the optimal node. The goal is to train a machine learning model to learn to rank the neighboring nodes based on each of the parameters, such that the order within the ranked list of nodes will represent the priority of the nodes. Because the relative order of any two nodes is not well defined, a model learns to approximate the order by modeling the correlations between the parameters and a score representing the position of the node within a set of examples ordered by an expert based on Table 2 and the priorities. This is known as a point-wise learning-to-rank approach.

Because the model should replicate the judgment of a human expert regarding which node should be selected as the next hop, building the dataset, essentially, becomes a matter of designing a protocol for interrogating that expert in order to extract as much useful knowledge as possible within the time an expert could be expected to give.

The solution involves first generating a list of 200 points from an eight-dimensional Sobol sequence spanning the unit hypercube. Unlike alternatives like randomly sampled points, using a Sobol sequence ensures that the parameter space is covered with a much lower discrepancy, thereby reducing the number of examples needed in the datasets. The list of points was then sorted using the Timsort algorithm, with each comparison being presented to the expert as a problem of determining which of the two nodes should be preferred. The parameter values were separated into three equal ranges and displayed using the text values "low," "medium," and "high." Using this simplified presentation of the raw values makes the questions easier for the expert to answer quickly, and for the low number of examples in our datasets, no two examples will be mapped to the same combination of text values. Finally, the target scores for each example were generated as the 0-indexed position within the ordered list and normalized to the range [0,1) by dividing the index by the total number of rows, with the lowest index being assigned to the worst node configuration.

TABLE 3 Definition of the search space used for XGBoost parameters

Hyperparameter	Mapping
Learning rate	x_1
Max depth	$2^{x_3 \log_2 8}$
Colsample bytree	x_4
Min child weight	$20.0x_5$
Subsample	x_6
Num round	$x_7(200.0 - 25.0) + 25.0$

The entire dataset is available in the comma-separated file[47] named "Machine Learning Dataset." In the file, the first eight columns contain the points in the parameter space that were generated from the Sobol sequence and used as input variables to the XGBoost algorithm. The rows are sorted according to the input from the expert, with the last column containing the normalized rank.

In order to learn a model to predict the experts' knowledge when presented with a previously unseen combinations of values, the XGBoost algorithm is used.[48] XGBoost is a state-of-the-art implementation of the popular gradient boosting decision tree algorithm, and is capable of both classification and nonlinear regression. XGboost is robust against noise in the data and can handle small datasets. The algorithm works by successively fitting many small decision tree learners with each new tree correcting the errors of the preceding trees. This makes the overall ensemble robust to overfitting, a significant problem with very small datasets.

The 10 most important parameters of the XGBoost algorithm were tuned by using a random search strategy that samples and evaluates 400 points from a uniform distribution spanning the $[1, 0]^6$ six-dimensional hypercube. These points are then mapped to the ranges required by the XGBoost algorithm by using the equations specified in Table 3, a mapping based on the approach in Reference 49. The best performing hyperparameters were then selected based on the mean absolute error (MAE) obtained from a 10-fold crossvalidation.

The model was evaluated on seven datasets, one using only the plain dataset values, and six containing from 1 to 6 of the principal components being added as additional predictors to the dataset. The hyperparameter procedure was repeated for each dataset, with only the best performing MAE from the 10-fold cross validation for each dataset being reported. The best performing parameters, which achieved a MAE of 0.10999, were a learning rate = 0.047197, max depth of 3.925077, colsample bytree = 0.728061, min child weight = 15.014327, subsample = 0.7615632, and num round = 170. The model used for the remainder of this article were fitted using these parameters to the entire dataset.

3.6 | Second chance method

When the node must choose between many resource-rich nodes within the local geographical area, all nodes will select the best node when using the method proposed in this article. That is because the method is designed to select the most suitable node, and in cases when there are several suitable nodes close to each other, our method, therefore, will select the best one. This can cause network congestion or inefficient resource utilization, as there are available resources in the local area, but all of the tasks are being executed on only one machine. Therefore, a second chance method that is designed to balance the load among the most suitable neighbors is also proposed in this article. The method consists of a fuzzy model that is used to compare the selected node with other neighbors in order to give a second chance to other suitable nodes of also being selected. This works by creating an array of all neighbors, sorted in descending order by score. Each node is then compared with the most suitable node. To compare the nodes selected by the choosing node N_i, called $\text{selected}_{\text{old}}(N_i)$, with other neighbors, the Equations (29) and (30) were used to calculate the inputs of the fuzzy method, as shown in Figure 2. In addition, Table 4 shows the rules of the fuzzy model, which as used in the second chance method, contains inputs and an output.

$$\text{Neighbor}_{\text{Diff}}(N_j) = \frac{\text{Neighbors}(N_i) - \text{selected}_{\text{old}}(N_i)}{\text{Neighbors}(\text{selected}_{\text{old}}(N_i))}, \tag{29}$$

$$\text{Score}_{\text{Diff}}(N_j) = \text{Score}(\text{selected}_{\text{old}}(N_i)) - \text{Score}(N_j). \tag{30}$$

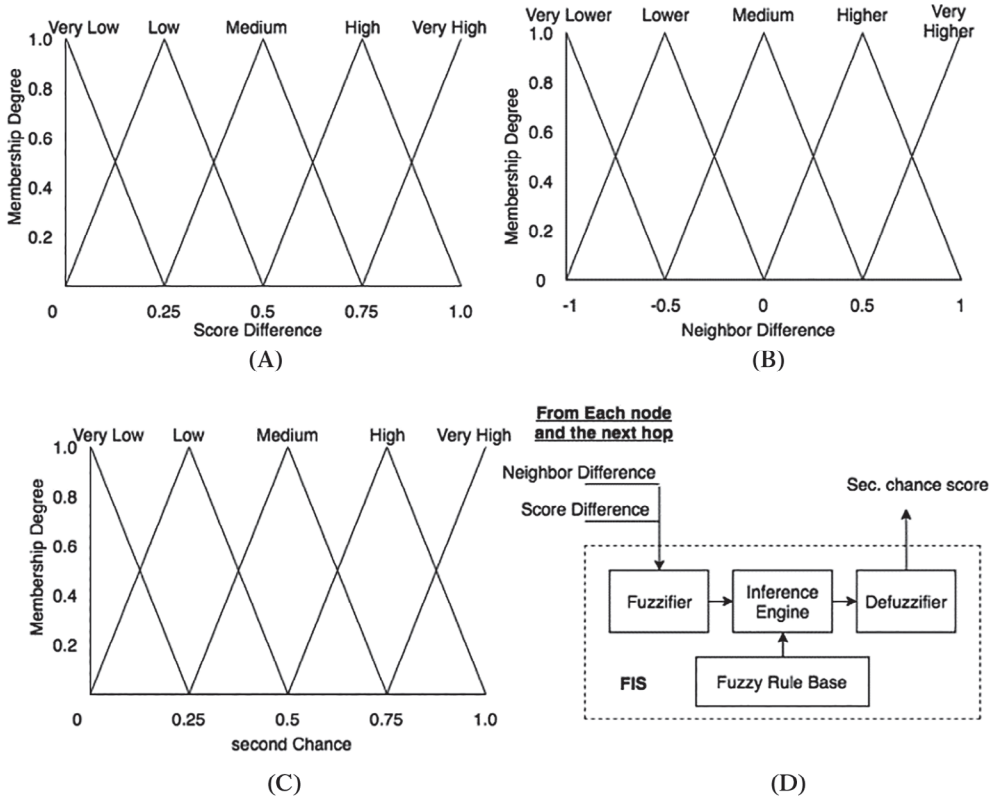


FIGURE 2 The fuzzy method that is used to give a second chance to other suitable nodes

Equation (31) shows the results of the second chance method based on Figure 2. As Score(N) returns the score of node N, $\text{Score}_{\text{second chance}}(N_j)$ returns the output of fuzzy method for Node N_j and Rand(x, y) generates a random float value between x and y and $\text{Selected}_{\text{new}}(N_i)$ shows the new eligible node which is selected by N_i.

$$\text{Selected}_{\text{new}}(N_i) = \begin{cases} N_j & \text{if Rand}(0, \text{Score}_{\text{second chance}}) + 1) > 1, \\ \text{Selected}_{\text{new}}(N_i) & \text{otherwise.} \end{cases} \tag{31}$$

The primary measures used for the second chance is the number of neighbors and scores, which are compared by using a fuzzy inference system (FIS) model that is described in Figure 2. In these figures, (A) and (B) are input variables of the FIS, (C) is the output of the FIS, and (D) shows the FIS model with input and output variables.

4 | PERFORMANCE EVALUATION

The performance of the proposed method was evaluated by modeling an IoT network consisting of 1024 nodes, which represents a general scenario. The types of nodes used are described in Table 5. In that table, the nodes are sorted based on their resources (energy and physical resources), and stipulate, for this scenario, that both the energy and hardware resources have the same priority.

The network is simulated using the Riverbed Modeler (formerly known as OPNET Modeler) version 18.5, as a commercial network modeler for modeling highly heterogeneous networks. The nodes are simulated using the "MANET Station advanced" node model available in OPNET, as the base class of IoT nodes in the simulation. The base model is extended by including an energy resource model. Additional network interfaces were added, in addition to the built-in network interface WLAN, in order to simulate a heterogeneous IoT ecosystem. In order to implement some of the logic into the simulation, the OPNET modeler connected to the MATLAB engine by using MX-Functions, such that OPNET calls the MATLAB engine. Each node sends its information to MATLAB engine and MATLAB returns the results to the modeler. In addition, to integrate the machine learning model, MATLAB calls Python3. From OPNET modeler to Python and vice versa, all tasks are suspended until the calls returned.

TABLE 4 Rules of fuzzy logic for second chance method

Rules			Rules		
$\text{Neighbor}_{\text{Diff}}$	$\text{Score}_{\text{Diff}}$	Second Chance Result	$\text{Neighbor}_{\text{Diff}}$	$\text{Score}_{\text{Diff}}$	Second Chance Result
Very lower	Very low	Very high	Medium	High	Very low
Very lower	Low	High	Medium	Very high	Very low
Very lower	Medium	Low	Higher	Very low	High
Very lower	High	Very low	Higher	Low	Medium
Very lower	Very high	Very low	Higher	Medium	Low
Lower	Very low	Very high	Higher	High	Very low
Lower	Low	High	Higher	Very high	Very low
Lower	Medium	Low	Very higher	Very low	Medium
Lower	High	Very low	Very higher	Low	Low
Lower	Very high	Very low	Very higher	Medium	Very low
Medium	Very low	High	Very higher	High	Very low
Medium	Low	Medium	Very higher	Very high	Very low
Medium	Medium	Very low			

To evaluate the performance of the proposed method, there were three types of results expected in selecting the most suitable nodes, as listed below:

- Resource-rich nodes will get a higher score when compared with their weak neighbors, and should then be selected. Neighbors and their circumstances can affect their scores, so, in most cases, the same types of nodes might not have the same score based on their local eligibility.
- In the presence of some resource-rich nodes in the geographical area, it was expected that the most resource-rich nodes received the highest scores.
- In the absence of resource-rich nodes, the most suitable weak node should get high score in its geographical area, but other weak nodes should get low scores.

The different scenarios are designed to test how good the proposed method is at selecting the most suitable nodes in a complex environment. In the absence of stronger nodes among neighbors, weaker nodes should be selected. Without using the second chance method, only the strongest nodes should be selected, but when the second chance is used, other relative strong nodes should also be selected. In Figures 3 to 7, two measurements were used to describe the generated scenarios:

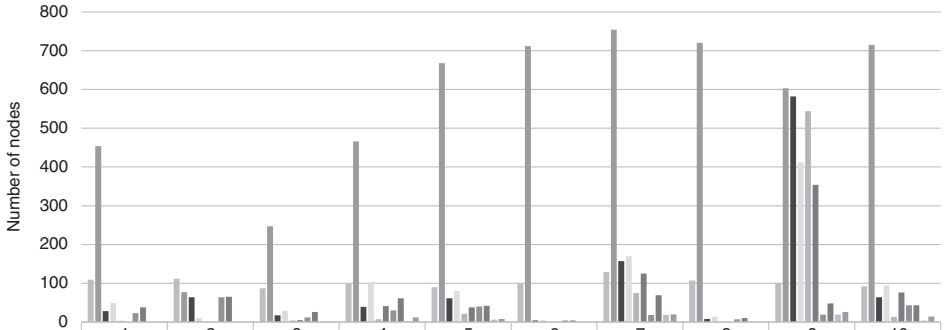
# of nodes: The number of nodes for that type
# of neighbors of the node type: How many nodes in the network can communicate with type x nodes include type x nodes.

In addition, eight measurements was also used to show how the proposed method selected the nodes in cases of both weak and resource-rich neighbors. A threshold parameter refined the number of selected nodes per type of node to show how nodes will converge to resource-rich nodes.

# of Clients: How many nodes selects nodes of type x as the most eligible nodes include type x nodes.
# of Clients (second chance): How many nodes selected nodes of type x by using the second chance method.
# of clients (Threshold $\geq$ 3): How many nodes are selected when each selected node has at least three clients.
# of clients (Second chance and Threshold $\geq$ 3): How many nodes select nodes of type x by using the second chance method when each selected node has at least three clients.

TABLE 5 Different type of nodes

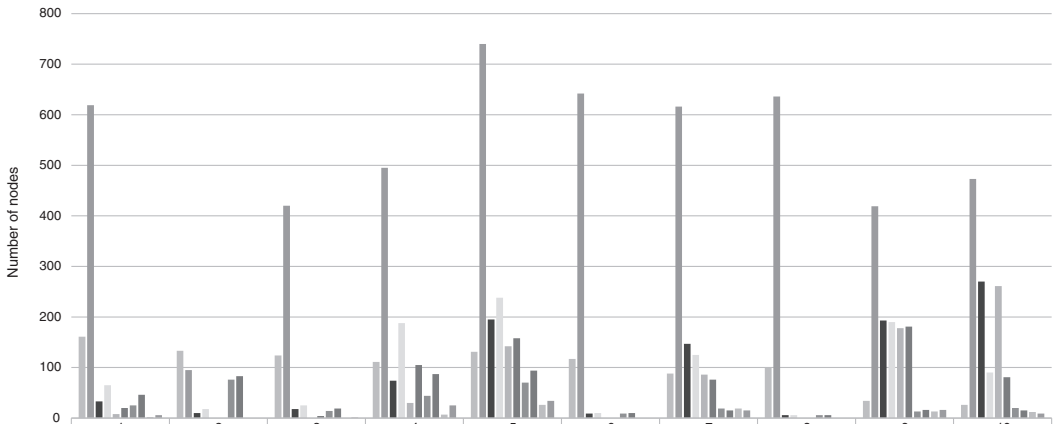
Node Type	Initial Energy (kJ)	CPU (MHz)	Storage (MB)	RAM (MB)	Applications								Network Interfaces					Node Example
					1	2	3	4	5	6	7	8	Bluetooth 3 Mbps 10 Meter	Zigbee 250 Kbps 30 Meter	WiFi 56 Mbps 50 Meter	WiFi-Direct 100 Mbps 150 Meter	Cellular D2D 150 Mbps 150 Meter	
1	7.4	16 MHz	Less than 1 MB	Less than 1 MB	*	*							*	*				Arduino Uno+ Bluetooth and Zigbee
2	1	1 GHz	4 GB	512 MB		*	*		*				*					Samsung Gear Fit 2
3	1.14	1.2 GHz	4 GB	512 MB			*		*				*		*			Huawei Watch1
4	22.1	700 MHz	512 MB	256 MB	*	*		*					*	*				Raspberry Pi A+
5	44.4	1.5 GHz	2 GB	4 GB	*			*		*	*		*	*	*			Raspberry Pi 4
6	9.03	1 GHz	8 GB	1 GB	*		*	*	*				*		*	*	*	Huawei Y3 II
7	94.03	2.2 GHz	128 GB	8 GB				*		*	*	*	*		*	*	*	Microsoft Surface G0 4G LTE
8	13.04	4.4 GHz	256 GB	4 GB	*		*		*	*		*	*		*	*	*	Samsung Galaxy S9
9	Infinite	5.4 GHz	8 GB	4 GB		*					*	*			*	*		Smart TV LG B8
10	165	4.5 GHz	1256 GB	8 GB				*		*	*	*	*		*	*		Lenovo ideapad L340 15 Gaming



	1	2	3	4	5	6	7	8	9	10
# of Nodes	109	112	87	99	90	98	129	107	101	92
# of Neighbors of the node type	454	77	247	466	668	712	754	720	603	715
# of Clients (General Method)	28	64	17	39	61	4	157	8	582	64
# of Clients (Sec. Chance)	49	10	29	103	80	5	170	14	412	94
# of Clients (Threshold>=3)	3	0	4	7	21	0	75	0	544	13
# of Clients (Sec. Chance and Threshold >=3)	0	0	5	41	38	0	125	0	354	76
# of selected nodes (General Method)	23	64	12	30	40	4	18	7	19	43
# of selected nodes (Sec. Chance)	38	65	26	61	42	4	69	10	48	43
# of selected nodes (General Method and Threshold>=3)	1	0	1	2	6	0	18	0	19	4
# of Selected nodes (Sec. Chance and Threshold>=3)	0	0	1	12	8	0	20	0	26	14

Type of node

FIGURE 3 Results of the proposed method for 1024 node and equal probability of node types



	1	2	3	4	5	6	7	8	9	10
# of Nodes	161	133	124	111	131	117	88	99	34	26
# of Neighbors of the node type	619	95	420	495	740	642	616	636	419	473
# of Clients (General Method)	33	10	18	74	195	9	147	6	193	270
# of Clients (Sec. Chance)	65	18	25	188	238	10	125	6	190	90
# of Clients (Threshold>=3)	8	0	0	30	142	0	86	0	178	261
# of Clients (Sec. Chance and Threshold >=3)	20	0	4	105	158	0	76	0	181	81
# of selected nodes (General Method)	25	76	14	44	70	9	19	6	13	20
# of selected nodes (Sec. Chance)	46	83	19	87	94	10	15	6	16	15
# of selected nodes (General Method and Threshold>=3)	2	0	0	7	26	0	19	0	13	12
# of Selected nodes (Sec. Chance and Threshold>=3)	6	0	1	25	34	0	15	0	16	9

Type of Node

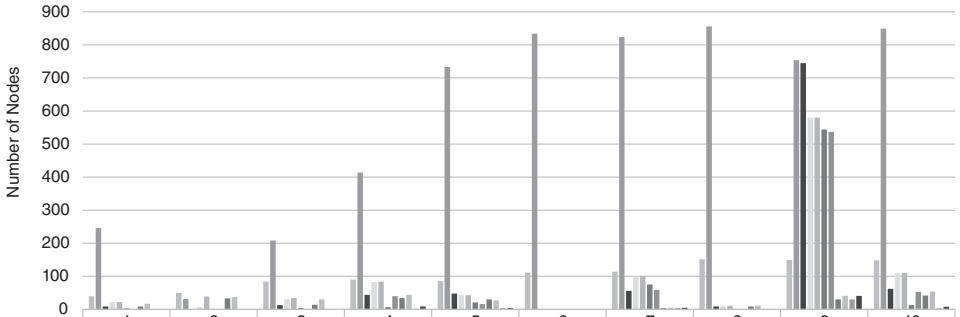
FIGURE 4 Results of the proposed method for 1024 node and higher chance of probability of weak nodes

# of selected nodes (General method): How many nodes of type x are selected without using second chance method.

# of selected nodes (Second chance): How many nodes of type x are selected by using the second chance method.

# of selected nodes (Threshold ≥ 3) How many of nodes of type x, are selected at least by three nodes.

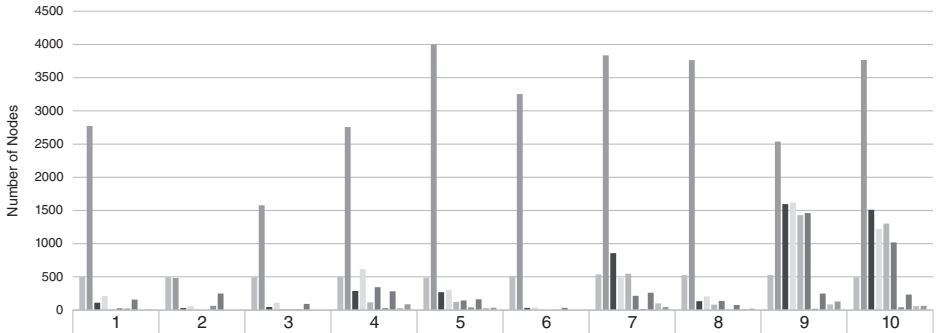
# of selected nodes (Second chance and Threshold ≥ 3) How many of nodes of type x are selected by at least three nodes.



	1	2	3	4	5	6	7	8	9	10
Type of Node	40	50	84	90	86	111	114	152	149	148
# of Nodes	246	31	208	414	733	834	824	856	754	849
# of Neighbors of the node type	9	1	13	44	48	1	56	9	745	62
# of Clients (General Method)	22	7	31	83	43	1	100	11	580	110
# of Clients (Sec. Chance)	22	39	34	84	43	1	100	11	580	110
# of Clients (Threshold>=3)	3	0	4	7	21	0	75	0	544	13
# of Clients (Sec. Chance and Threshold >=3)	0	0	0	40	16	0	59	0	537	53
# of selected nodes (General Method)	9	33	14	34	30	1	4	9	30	42
# of selected nodes (Sec. Chance)	17	37	30	44	27	1	5	11	41	54
# of selected nodes (General Method and Threshold>=3)	0	0	1	2	4	0	4	0	30	3
# of Selected nodes (Sec. Chance and Threshold>=3)	0	0	0	9	4	0	5	0	41	8

Type of Node

FIGURE 5 Results of the proposed method for 1024 node and higher chance of probability of resource-rich nodes



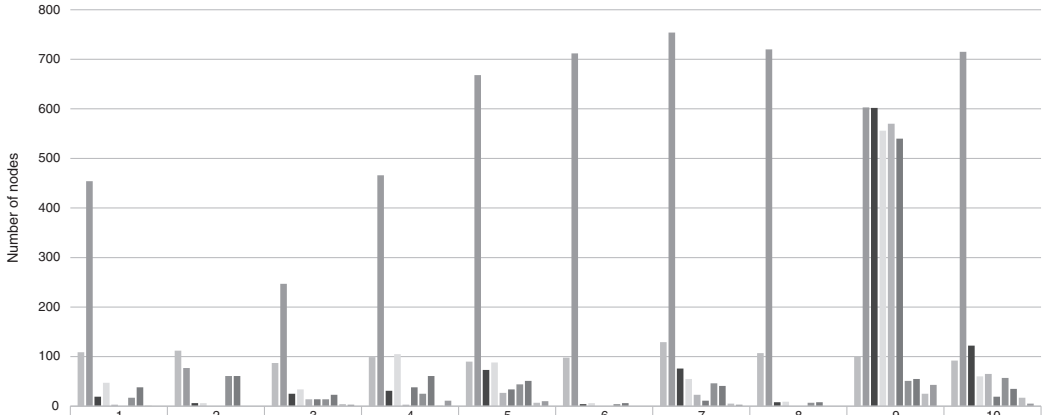
	1	2	3	4	5	6	7	8	9	10
# of Nodes	504	494	493	500	487	513	537	526	529	493
# of Neighbors of the node type	2771	484	1579	2756	4003	3254	3835	3765	2537	3765
# of Clients (General Method)	112	31	45	289	269	34	858	134	1596	1510
# of Clients (Sec. Chance)	209	59	112	619	306	37	487	206	1619	1224
# of Clients (Threshold>=3)	9	0	13	116	124	4	548	82	1430	1302
# of Clients (Sec. Chance and Threshold >=3)	28	0	9	345	144	0	216	137	1462	1020
# of selected nodes (General Method)	23	64	12	30	40	4	18	7	19	43
# of selected nodes (Sec. Chance)	158	250	94	283	163	32	263	78	248	232
# of selected nodes (General Method and Threshold>=3)	3	0	3	31	31	1	100	14	85	61
# of Selected nodes (Sec. Chance and Threshold>=3)	8	0	3	87	36	0	46	21	129	64

Type of node

FIGURE 6 Results of the proposed method for 5076 nodes and equal probability

In some cases, the sum of "# of nodes" for all node types is lower than the total nodes contributed in each simulation, for example, 1024 in scenario 1. This is because the nodes are randomly distributed, and some nodes have no neighbors in the area covered with at least one similar network interface. These nodes are not counted in the results, as they have no neighbors to compare or be compared against.

As the method is designed to be used in IoT networks, many nodes are required to show whether it is efficient or not. Therefore, statistical information on the suitability scores and their correlations with the different types of nodes and the types of neighbors of each node, was summarized, as it is impossible to show results in detail for more than 1000 nodes. The tables help to understand the behavior of the method in the various scenarios. In Tables 6, 7, 8, 10, 11, the columns are based on the average number of node types of each node separated by their scores. As an example, in Table 6, the



	1	2	3	4	5	6	7	8	9	10
# of Nodes	109	112	87	99	90	98	129	107	101	92
# of Neighbors of the node type	454	77	247	466	668	712	754	720	603	715
# of Clients (General Method)	19	6	25	31	73	4	76	8	602	122
# of Clients (Sec. Chance)	47	6	34	105	88	6	55	9	556	60
# of Clients (Threshold>=3)	3	0	14	3	27	0	23	0	570	65
# of Clients (Sec. Chance and Threshold >=3)	0	0	14	38	34	0	11	0	540	19
# of selected nodes (General Method)	17	61	14	25	44	4	46	7	51	57
# of selected nodes (Sec. Chance)	38	61	23	61	51	6	41	8	55	35
# of selected nodes (General Method and Threshold>=3)	1	0	4	1	7	0	5	0	25	17
# of Selected nodes (Sec. Chance and Threshold>=3)	0	0	3	11	10	0	3	0	43	5

Type of Nodes

FIGURE 7 The results of the fuzzy model

TABLE 6 Average types of neighbors for different types of nodes of scenario 1

Type of Nodes	0,1=<	0,1> & <=0,2	0,2> & <=0,3	0,3> & <=0,4	0,4> & <=0,5	0,5> & <=0,6	0,6> & <=0,7	0,7> & <=0,8	0,8> & <=0,9	0,9> & <=1.0
Type 1	3.53	0	5.45	5.81	4.56	3.00	0	0	0	0
Type 2	0	0	3.00	6.02	1.00	2.00	0	0	0	0
Type 3	3.27	3.57	4.33	0	0	3.00	0	0	0	0
Type 4	0	0	6.11	8.02	5.16	4.14	2.00	0	0	0
Type 5	0	6.20	6.85	0	6.40	5.58	4.33	0	0	0
Type 6	7.76	7.76	7.85	0	0	0	0	0	0	0
Type 7	0	7.96	0	0	0	7.73	7.73	0	0	0
Type 8	5.76	6.76	4.99	0	0	0	0	0	0	0
Type 9	0	0	0	0	7.91	7.88	7.86	0	0	0
Type 10	0	0	0	0	0	7.72	7.72	0	0	0

value of the first column and the first row means that nodes which are an instance of Type 1 and their scores are between 0 and 0.1, have the average neighborhood of type 3.53. For node type t and $0 < \text{score}\{N\} < 0.1$, used Equation (33). In addition, 0 in the tables means that there are no nodes of type t with the aforementioned score. The tables can help to show the scores that the proposed method provides the nodes in order to compare their neighbors. Furthermore, all of the information that is needed to mimic the scenarios in various datasets, is provided. Datasets related to the scenarios in the performance evaluation section are available in Reference 47 in a CSV format that contains the node name, selected eligible node by each node, the node type, the number of clients per node, the number of neighbors, the score, the average type of neighbors per node and position of nodes (metrics of node position is in kilometers).

$$\text{Avg}_{\text{Neighbor\_ Types}}(N_i) = \frac{\sum_{n=j}^{z} \text{Type}(N_n)}{C}, \tag{32}$$

TABLE 7 Average types of neighbors for different types of nodes of scenario 2

Type of Nodes	0,1=<	0,1> & <=0,2	0,2> & <=0,3	0,3> & <=0,4	0,4> & <=0,5	0,5> & <=0,6	0,6> & <=0,7	0,7> & <=0,8	0,8> & <=0,9	0,9> & <=1.0
Type 1	2.80	3.52	4.79	4.69	4.01	1.00	0	0	0	0
Type 2	0	0	3.42	5.28	2.50	2.00	0	0	0	0
Type 3	3.69	3.44	4.50	0	0	2.94	0	0	0	0
Type 4	0	0	5.45	4.82	4.77	3.55	2.88	0	1.00	0
Type 5	0	5.78	5.89	6.87	5.69	4.92	4.67	0	0	0
Type 6	7.07	7.02	7.06	0	0	0	0	0	0	0
Type 7	0	0	0	0	0	7.03	6.50	0	0	0
Type 8	6.93	7.11	0	0	0	0	0	0	0	0
Type 9	0	0	0	0	7.44	7.31	7.20	0	0	0
Type 10	0	0	0	0	0	7.20	6.96	0	6.89	0

TABLE 8 Average types of neighbors for different types of nodes of scenario 3

Type of Nodes	0,1=<	0,1> & <=0,2	0,2> & <=0,3	0,3> & <=0,4	0,4> & <=0,5	0,5> & <=0,6	0,6> & <=0,7	0,7> & <=0,8	0,8> & <=0,9	0,9> & <=1.0
Type 1	0	4.00	6.10	4.50	5.63	0	0	0	0	0
Type 2	0	0	1.00	6.80	0	0	0	0	0	0
Type 3	3.50	3.89	4.00	0	0	3.00	0	0	0	0
Type 4	0	0	6.75	6.54	5.65	4.00	3.65	0	0	0
Type 5	0	0	7.21	0	6.78	4.88	3.75	0	0	0
Type 6	8.06	8.05	7.76	0	0	0	0	0	0	0
Type 7	0	8.38	0	0	8.25	8.02	8.13	0	0	0
Type 8	0	8.08	7.93	0	0	0	0	0	0	0
Type 9	0	0	0	0	8.05	8.16	8.14	0	0	0
Type 10	0	0	0	0	0	8.02	8.24	0	0	0

TABLE 9 Comparing Scores of node types in different Scenarios

	Lowest Scores			Average Scores			Highest Scores		
Node Type	Scenario #1	Scenario #2	Scenario #3	Scenario #1	Scenario #2	Scenario, #3	Scenario #1	Scenario #2	Scenario #3
Type 1	0,080682	0,091326	0,101136	0,280619	0,273788	0,281995	0,525593	0,558279	0,462792
Type 2	0,230998	0,249944	0,244217	0,360781	0,339749	0,326356	0,585003	0,578254	0,382031
Type 3	0,094999	0,092158	0,093005	0,187139	0,1517	0,154855	0,566587	0,579859	0,586882
Type 4	0,235371	0,236394	0,244702	0,372241	0,446135	0,352583	0,642607	0,837473	0,669015
Type 5	0,11599	0,152482	0,221638	0,33742	0,457884	0,283445	0,659265	0,628911	0,663337
Type 6	0,073004	0,066568	0,078809	0,112001	0,103326	0,114175	0,233834	0,236155	0,234692
Type 7	0,152914	0,513605	0,151778	0,572312	0,548443	0,561509	0,625307	0,623713	0,622107
Type 8	0,091127	0,077896	0,10075	0,131941	0,110897	0,135595	0,249481	0,139561	0,251711
Type 9	0,484726	0,471886	0,472975	0,611428	0,603241	0,608705	0,644681	0,638673	0,659195
Type 10	0,525314	0,505227	0,547384	0,574583	0,577496	0,58624	0,624681	0,801077	0,627868

TABLE 10 Average types of neighbors for different types of nodes of scenario 4

Type of Nodes	0,1=<	0,1> & <=0,2	0,2> & <=0,3	0,3> & <=0,4	0,4> & <=0,5	0,5> & <=0,6	0,6> & <=0,7	0,7> & <=0,8	0,8> & <=0,9	0,9> & <=1.0
1	3.70	3.13	5.53	6.72	4.56	3.33	0	0	0	0
2	0	0	5.99	5.93	2.65	2.00	2.00	0	0	0
3	3.48	3.61	4.39	0	0	3.00	0	0	0	0
4	0	0	6.04	6.41	5.10	4.47	3.08	0	1.00	0
5	0	0	6.63	6.03	6.16	5.81	0	0	0	0
6	7.50	7.56	7.49	7.53	7.55	7.56	7.57	0	0	0
7	0	8.07	0	0	7.83	7.76	7.90	7.85	0	0
8	7.35	7.80	7.68	0	0	7.77	7.86	7.84	0	0
9	0	0	0	0	0	8.36	8.36	8.28	0	0
10	0	0	0	0	0	7.78	8.00	7.83	7.89	0

TABLE 11 Average types of neighbors for different types of nodes of scenario 5

Type of Nodes	0,1=<	0,1> & <=0,2	0,2> & <=0,3	0,3> & <=0,4	0,4> & <=0,5	0,5> & <=0,6	0,6> & <=0,7	0,7> & <=0,8	0,8> & <=0,9	0,9> & <=1.0
1	0	6.95	5.63	4.07	4.50	3.00	0	0	0	0
2	0	6.63	1.63	1.50	2.71	0	0	0	0	0
3	0	3.68	2.49	2.40	2.96	0	0	0	0	0
4	0	10.00	7.01	5.57	4.67	2.54	0	0	0	0
5	0	0	7.47	6.70	6.19	4.00	0	4.33	0	0
6	0	7.75	7.86	0	0	0	0	0	0	0
7	0	0	0	0	7.72	7.77	0	0	0	0
8	0	7.74	7.75	0	0	0	0	0	0	0
9	0	0	0	0	0	7.90	7.85	7.36	0	0
10	0	0	0	0	7.83	7.70	0	0	0	0

$$\mathrm{Avg}_{\mathrm{Types}}(\mathrm{Nodes}_N(0 < \mathrm{Score}(N_{\mathrm{Type}_i}) \leq 0.1)) = \frac{\mathrm{Sum}_{(\mathrm{Nodes}(0<\mathrm{Score}(N_{\mathrm{Type}_t})\leq 0.1))}(\mathrm{Avg}_{\mathrm{Neighbor\_Types}}(N_i))}{\mathrm{Count}(\mathrm{Nodes}(0 < \mathrm{Score}(N_{\mathrm{Type}_t}) \leq 0.1))}. \tag{33}$$

Here, $\mathrm{Avg}_{\mathrm{Neighbor\_Types}}(N_i)$ is the average types of N_i neighbors and $\mathrm{Sum}_r(v)$ is sum of v for nodes that satisfy condition r. In addition, $\mathrm{Count}(r)$ is the number of nodes that satisfy condition r.

4.1 | Scenario 1: Equal distribution of all types of nodes

This scenario is built by randomly placing 1024 nodes in a 200 × 200-meter environment. To assign a node type to each node, samples was drawn from a uniform probability distribution, such that a node as an equal chance to be one of any of the ten node types. Figure 3 shows the distribution of the different node types in the randomly generated scenario.

When comparing the # of neighbors with the # of clients, the most selected nodes are of type 9 and 7 when not using the second chance method, as they have the highest energy resources. When the second chance method is used, the distribution of the clients is much better at also selecting weaker, but the still usable, type 4 and 5 nodes as 220% and 30% increase, respectively. In addition show in the figure, without using the second chance method, more than 55% of the nodes connect to the type 9, but by using the second chance method, this is decreased to 40%. The number of nodes that select the type 10 nodes is also increased to 40%. The figure also shows that although weak nodes are selected rarely, by using the threshold, the weak nodes can be deselected when they are in an area in which all the other neighbors are not

weak, as well and they only selected based on only having few weak nodes. The threshold can help to solve the problem of selecting weak nodes in sparse areas, as in some cases weak nodes have only few neighbors at the edge of the network which cannot connect to other nodes as nodes in blind spaces. If they are in an area that all nodes are weak, then more than three nodes should select them as eligible nodes. Otherwise, they will be deselected based on the threshold. As an example, in scenario 1, the nodes of type 1, 2, and 3 are sometimes selected, but when using the threshold, most of them will be deselected. Besides, it shows that type 8 is not selected because of its low energy resources, even when having excellent physical resources or circumstances when compared with its neighbors. As shown in the figure, most of the weak nodes have only a few clients, as shown by "# of selected nodes" for type 1 being 23, and the number of clients is 28, and 22 of them are deselected by using the threshold. On the other hand, in resource-rich nodes, the number of clients has mostly remained the same, especially when the second chance method and a threshold are used simultaneously. As an example, nodes of type 7 has 157 clients. And by using threshold and the second chance method, it decreases to 125. This shows that the second chance method can reselect the resource-rich nodes that are deselected by threshold and avoid reselecting weak nodes.

Table 6 shows the average types of neighbors for the different types of nodes. As shown in the table, most of the nodes, which have a lower score than the same types of nodes, have neighbors who are richer in terms of different types of resources. In addition, in other cases, the nodes have a higher score when compared with the types of their neighbors because of their circumstances. In this article, when a node has a better circumstance when compared with its neighbors means that the node has better circumstances in terms of application and physical layer parameters. Since its same probability of all types of nodes, the neighbors of node N_i can be very diverse in terms of their types. While this does put the proposed method under pressure, it can still handle nodes, such as type 6 and beyond, and most of the nodes, which have strong neighbors, still have a high score. As expected, there are some weak nodes that have a relatively high score because they have only weak neighbors, but none of them are more than 0.5, which shows that being in a very well situation cannot overcome the resource parameters which have the highest priorities. In addition, the table shows that although type 8 has excellent resources, the low energy resource parameter means that they cannot overcome other types of nodes and, therefore, only get low scores.

4.2 | Scenario 2: Higher proportion of weak nodes

In this scenario, the number of weak nodes is higher than resource-rich nodes. To assign different types to the nodes, a random value is generated for each node by $\text{Type}_{\text{Rand}}(N_i)$, which is in $(0, 1)$. Then, the generated value is searched for in P and N_i is an instance of type x node when $1 \leq x < 10$ and $2 < y \leq 10$, and also they satisfied : Condition 1: $P[x] \leq \text{Type}_{\text{Rand}}(N_i) \leq P[y]$ and Condition 2: $y - x = 1$. For this scenario, $P = [0, 0.15, 0.29, 0.42, 0.54, 0.65, 0.75, 0.84, 0.93, 0.97, 1.0]$. The results for this scenario are shown in Figure 4. As shown in the figure, most of the selected nodes are of type 10, 9, 7, or 5. In cases of having less resource-rich nodes, node type 10 is selected more than scenario 1 equal probability as less Type 9 and Type 10 nodes are close to each other. In addition, nodes Type 5 are selected three times more than scenario 1 as they are more available than other type of resource-rich nodes in the network. With the second chance method, there is a chance to select nodes of type 4 as they are, relatively, more suitable and available than the other types. The load of clients on Type 10 is decreased to 60%, because of the added other types. As shown in the figure, the number of clients of type 4 nodes increases to about 150%, and the number of clients of type 5 nodes increases to 25%. Without using a threshold, the number of clients of type 1 is high, but by using the threshold, some weak nodes are still selected as eligible nodes as the average types of neighbors is not too high, as shown in Table 7, but most them are deselected, which shows that few nodes selected them, initially. In addition, as shown in the table, in some cases, although some nodes have fewer resources, they still get high scores because the situation compares favorably to that of their neighbors. In addition, Figure 4 shows that using the second chance method and threshold simultaneously gives a higher chance of selecting weak nodes to balance the load, which means that each weak node only needs to have a few clients. This ability to balance the load among multiple nodes becomes essential when there is a shortage of resource-rich nodes, as each weak node is only capable of offering their resources to a few neighbors.

4.3 | Scenario 3: Higher proportion of resource-rich nodes

In this scenario, a higher proportion of strong nodes is generated using the same procedure described in Section 4.2, but with $P = [0, 0.04, 0.08, 0.16, 0.25, 0.35, 0.46, 0.58, 0.71, 0.85, 1.0]$.

As shown in the figure, in this instance, most of the selected nodes are of type 9, as they are available everywhere (more than 15% of nodes), and because they have high energy resources. Furthermore, the other resource-rich nodes are not likely to be selected by neighbors without using the second chance method. By using the second chance method, some nodes select other resource-rich nodes, as shown by the number of clients of type 4 and 7 which increased by about 100%, type 10 which increased by about 90%, and clients of node Type 9 which instead decreased by about 25%. In addition, using the threshold shows that all weak nodes were selected by few clients, which are deselected. In addition, Table 8 shows that, having a lot of resource-rich nodes, the scores of the nodes decreases since most resource-rich nodes are between 0.4 and 0.7, and most of the weak nodes are between 0.1 and 0.4. The types of neighbors have less impact on the scores when compared with Tables 6 and 7, as nodes in the current scenario are much stronger than previous scenarios, but the correlation between the scores of the weak nodes and those of the resource-rich nodes is still set. Moreover, Figure 5 shows that most of the weak nodes are selected more rarely than three nodes; nevertheless, most of the resource-rich nodes are selected by more than three nodes, so there are no tangible changes in the number of selected nodes when using the threshold but the second chance method helps balance the load between them.

4.3.1 | Comparing different proportions

To compare the scenarios and the impact of the types of nodes on the scores, Table 9 is used. The table compares the minimum, maximum, and average scores of nodes in different scenarios. As shown in the table, the lowest scores are, in most cases, sensitive to change in the contribution of different types of nodes. In scenario 3, the highest scores are reached when compared with other scenarios in case of having the lowest score, as shown by having more of the resource-rich nodes increased the score of all nodes. For average scores, scenario 1 has the highest average scores, which shows that having the same contribution of types of nodes can conceal most of the suitable nodes as the average scores are close to highest scores, and it can make the selection method more complicated. In addition, for the highest scores, the first two scenarios have approximately the same values, but scenario 3 has lower scores, which shows that having more resource-rich nodes can decrease scores but also reveal the most eligible nodes as highest and average scores are not close. Overall, by comparing the difference between the lowest, average, and highest scores of each type of node shows that scenario 1 has the highest diversity of scores for each type of node, as expected. Scenario 3 has the lowest diversity, which means that having more resource-rich nodes can make the selection process more complicated as scores are closer to each other and can conceal the neighbors that would be suitable. As shown in Table 8, the second chance method can help to solve the problem by giving a second chance to nodes that are also suitable. In addition, in case of having same probability for all nodes, having threshold can help to deselect weak nodes, which have received high scores because of their mixed types of neighbors.

4.4 | Scenario 4: UDN IoT network

To assess the performance of the method in Massive-scale UDN IoT networks, a scenario consisting of 5076 nodes in a 300×300 meters environment was designed, generated using the same method as scenario 1. The details are available in Reference 47.

Equation 6 shows the results of the proposed method for scenario 4. As shown in the figure, although the weak nodes as part of a UDN have more neighbors within their coverage area, they still have the most resources and better circumstances compared with their neighbors. Because a mIoT network has more nodes, the average types of neighbors, when compared with their scores, are increased, as shown in Table 10. Unlike the other scenarios, some nodes have scored higher than 0.7, and this shows that the density of the network can affect the proposed method and the scores. But the correlation between the scores of weak and resource-rich nodes in two different scenarios with different numbers of nodes present. Most of the weak nodes have also not attained scores better than 0.5, even when the circumstances are better than their neighbors. In addition, as shown in Figure 6, there are no tangible changes in the number of clients for resource-rich nodes when a threshold is used. The figure also demonstrates, as for general networks, that the selected weak nodes only have a few clients, and a threshold can be used to remove most of them.

4.5 | Scenario 5: Comparing the proposed method

For this scenario, the model introduced in Reference 50 was used, which is based on a fuzzy method to compare the IoT nodes. In this model, all possible rules of the fuzzy model are extracted by a learning-to-rank method to compare nodes.

The results of using the method introduced in Reference 50 are shown in Figure 7. As shown in the figure, the distribution of the selected nodes among resource-rich nodes is more unfair than scenario 1, resulting from having more of the same types of nodes. Almost all nodes which are within the coverage area of a type 9 node and, therefore, selects them. By using the second chance method, the distribution of selected nodes is fairer, but still, the number of nodes that select type 9 is high because the scores are so close to each other. Besides, using a threshold can show that the node assessment model, based on fuzzy, selects more weak nodes. The figure also shows that, compared with scenario 1, nodes of Type 4, 5, 7, and 10, that are resource-rich, are selected less, even by using the second chance method. It confirms that the scores which are generated by the fuzzy-based method are closer to each other, which makes the selection procedure complicated and sensitive. In addition shown in Table 11, is the results of the average type of neighbors for the fuzzy-based method. The distribution of scores in the proposed method is better than the fuzzy-based method, as most of the nodes have scores between 0.1 and 0.6, but the scores in the proposed method is more evenly distributed between 0.0 and 0.8, which lessens the chance of selecting an inefficient node.

As a conclusion, performance evaluation section shows that the proposed method is able to reveal eligible nodes in mIoT networks with thousands of nodes. The proposed method is able to detect nodes which are in a good situation and have more resources than their neighbors. When most of the nodes are weak, the method selects the nodes which are better than their neighbors. On the other hand, when there are considerable number of strong nodes are available, the proposed method helps to detect the strongest. In addition, the second chance method help the pure method to balance the number of members of eligible nodes in case of having several strong nodes close to each other. In addition, the tables show that score of nodes in different cases is related to neighbors, but consequently selected nodes are eligible globally even in UDN to dedicate their resources to neighbors in one-hop communications.

5 | CONCLUSION AND FUTURE WORKS

Recently, computing paradigms like Edge and Mist are aiming to move more and more tasks and services closer to the end users. However, this requires identifying nodes that are able to share their resources, which can be a complicated task, especially in mIoT networks, as the huge number of nodes can hide the most suitable candidates. In this article, a novel method for evaluating the nodes in a highly heterogeneous-mIoT network is proposed that helps to identify relative resource-rich nodes. When evaluated in simulations of five different scenarios, the method succeeds in identifying the powerful nodes, when those nodes are available, while in absence resource-rich nodes, the local and relative resource-rich node are selected.

While, there are a few evaluation methods proposed in the literature, existing methods do not consider several layers requirements during the nodes comparison, such as requirements from the application and physical layers that are needed for the methods to adapt to networks such as massive-scale and highly heterogeneous IoT. In addition, as the methods do not consider different layers requirements, selected nodes are not optimally able to serve tasks like caching, data aggregation, or executing containers. The method proposed in this article considers applications, hardware resources, the situation of the neighbors and environmental factors in the comparison of nodes to identify most suitable nodes, thereby increasing the effectiveness of techniques aiming to move computation closer to the end users.

The proposed method works by assessing the usefulness of neighboring nodes by comparing scores received from them. The score of each node is calculated by comparing its information and its situation with its neighbors. The comparison considers aspects like hardware resources, network interfaces, and the number of neighbors, all of which are then combined to form a single final floating-point score by using a learning-to-rank method. Because there can be several resourceful nodes in the network, a novel load balancing technique, named second chance, based on fuzzy logic, is also proposed. This method is used to reevaluate the selection of a node in order to also select other, only slightly less resourceful nodes.

The proposed method was evaluated using simulations of five scenarios designed to test different situations of networks including massive-scale networks and different quantities of resource-constraint and resource-rich nodes. In all scenarios, the proposed methods succeed in selecting the most suitable nodes, even in the scenario representing UDN networks. When evaluating the second chance method, other powerful nodes were also selected, thereby balancing the load among the most powerful nodes. But the much less powerful nodes remained largely unselected.

The main idea of the article is that the proposed method will form the foundation of a new clustering technique to unlock the power of highly heterogeneous IoT-Mist networks. The method can be further extended for use in a cluster head selection method to manage and share the available resources in IoT-Mist networks, but this is considered as future

work. In addition, the method can be used to develop distributed computing in massive-scale networks, as it is able to identify the most suitable nodes to contribute to the computation. The proposed method is vulnerable to security attacks involving a malicious actor broadcasting false information about their resources and situation. Avoiding such security attacks is considered as future work as well.

ORCID

Amin Shahraki https://orcid.org/0000-0002-5065-9968

REFERENCES

1. Whitmore A, Agarwal A, Da Xu L. The internet of things-A survey of topics and trends. *Inf Syst Front*. 2015;17(2):261-274.
2. Scuotto V, Ferraris A, Bresciani S. Internet of things: applications and challenges in smart cities: a case study of IBM smart city projects. *Bus Process Manag J*. 2016;22(2):357-367.
3. Marques G, Garcia N, Pombo N. A survey on IoT: architectures, elements, applications, QoS, platforms and security concepts. *Advances in Mobile Cloud Computing and Big Data in the 5G Era*. New York, NY: Springer; 2017:115-130.
4. Biswas AR, Giaffreda R. IoT and cloud convergence: opportunities and challenges. Paper presented at: Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT); 2014:375-376; IEEE.
5. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. *IEEE Internet Things J*. 2016;3(5):637-646.
6. Porambage P, Okwuibe J, Liyanage M, Ylianttila M, Taleb T. Survey on multi-access edge computing for internet of things realization. *IEEE Commun Surv Tutor*. 2018;20(4):2961-2991.
7. Fernando N, Loke SW, Rahayu W. Mobile cloud computing: a survey. *Futur Gener Comput Syst*. 2013;29(1):84-106.
8. Dastjerdi AV, Buyya R. Fog computing: helping the internet of things realize its potential. *Computer*. 2016;49(8):112-116.
9. Davies A. Cisco pushes iot analytics to the extreme edge with mist computing; 2014.
10. Vasconcelos D, Andrade R, Severino V, De Souza JN. Cloud, fog, or mist in IoT? that is the question. *ACM Trans Internet Tech (TOIT)*. 2019;19(2):25.
11. George AM, Kulkarni S, George V. A survey on ultra low power design techniques for IOT application. *Current Trends Inf Tech*. 2018;7(3):9-16.
12. Musaddiq A, Zikria YB, Hahm O, Yu H, Bashir AK, Kim SW. A survey on resource management in IoT operating systems. *IEEE Access*. 2018;6:8459-8482.
13. Elazhary H. Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: disambiguation and research directions. *J Netw Comput Appl*. 2018;128:105-140.
14. Samie F, Bauer L, Henkel J. IoT technologies for embedded computing: a survey. Paper presented at: Proceedings of the 11th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis; 2016:8; ACM.
15. Sehgal A, Perelman V, Kuryla S, Schonwalder J. Management of resource constrained devices in the internet of things. *IEEE Commun Mag*. 2012;50(12):144-149.
16. El-Sayed H, Sankar S, Prasad M, et al. Edge of things: the big picture on the integration of edge, IoT and the cloud in a distributed computing environment. *IEEE Access*. 2017;6:1706-1717.
17. Yousefpour A, Fung C, Nguyen T, et al. All one needs to know about fog computing and related edge computing paradigms: a complete survey. *J Syst Archit*. 2019;10:289-330.
18. Mahmood Z, Ning H, Ghafoor A. Lightweight two-level session key management for end user authentication in Internet of Things. Paper presented at: Proceedings of the 2016 IEEE International Conference on Internet of Things (iThings); 2016:323-327; IEEE.
19. Parwekar P. From internet of things towards cloud of things. Paper presented at: Proceedings of the 2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011); 2011:329-333; IEEE.
20. Xiao G, Guo J, Da Xu L, Gong Z. User interoperability with heterogeneous IoT devices through transformation. *IEEE Trans Ind Inf*. 2014;10(2):1486-1496.
21. Bandyopadhyay D, Sen J. Internet of things: applications and challenges in technology and standardization. *Wirel Pers Commun*. 2011;58(1):49-69.
22. Yu W, Xu H, Zhang H, Griffith D, Golmie N. Ultra-dense networks: survey of state of the art and future directions. Paper presented at: Proceedings of the 2016 25th international conference on computer communication and networks (ICCCN); 2016:1-10; IEEE.
23. Datta SK, Da Costa RPF, Bonnet C. Resource discovery in Internet of Things: current trends and future standardization aspects. Paper presented at: Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT); 2015:542-547; IEEE.
24. Torkestani JA. A distributed resource discovery algorithm for P2P grids. *J Netw Comput Appl*. 2012;35(6):2028-2036.
25. Chowdhury A, Raut SA. A survey study on internet of things resource management. *J Netw Comput Appl*. 2018;120:42-60.
26. Meshkova E, Riihijärvi J, Petrova M, Mähönen P. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Comput Netw*. 2008;52(11):2097-2128.
27. Perera C, Vasilakos AV. A knowledge-based resource discovery for Internet of Things (IoT). *Knowl-Based Syst*. 2016;109:122-136.
28. Hong CH, Varghese B. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM Comput Surv (CSUR)*. 2019;52(5):97.
29. Wang L. SoFA: an expert-driven, self-organization peer-to-peer semantic communities for network resource management. *Expert Syst Appl*. 2011;38(1):94-105.

30. Merz P, Gorunova K. Fault-tolerant resource discovery in peer-to-peer grids. *J Grid Comput*. 2007;5(3):319-335.
31. Arunachalam A, Sornil O. Issues of implementing random walk and gossip based resource discovery protocols in P2P MANETs & suggestions for improvement. *Proc Comput Sci*. 2015;57:509-518.
32. Talpur MSH, Bhuiyan MZA, Wang G. Shared–node IoT network architecture with ubiquitous homomorphic encryption for healthcare monitoring. *Int J Embed Syst*. 2014;7(1):43-54.
33. Abedin SF, Alam MGR, Tran NH, Hong CS. A fog based system model for cooperative IoT node pairing using matching theory. Paper presented at: Proceedings of the 2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS); 2015:309-314; IEEE.
34. Aazam M, Huh EN. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. Paper presented at: Proceedings of the 2015 IEEE 29th International Conference on Advanced Information Networking and Applications; 2015:687-694; IEEE.
35. Zhou A, Wang S, Li J, Sun Q, Yang F. Optimal mobile device selection for mobile cloud service providing. *J Supercomput*. 2016;72(8):3222-3235.
36. Fernando N, Loke SW, Rahayu W. Computing with nearby mobile devices: a work sharing algorithm for mobile edge-clouds. *IEEE Trans Cloud Comput*. 2016;7(2):329-343.
37. Elahi BM, Romer K, Ostermaier B, Fahrmair M, Kellerer W. Sensor ranking: a primitive for efficient content-based sensor search. Paper presented at: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks IEEE Computer Society; 2009:217-228.
38. Preden JS, Tammemäe K, Jantsch A, Leier M, Riid A, Calis E. The benefits of self-awareness and attention in fog and mist computing. *Computer*. 2015;48(7):37-45.
39. Barik RK, Tripathi A, Dubey H, et al. Mistgis: optimizing geospatial data analysis using mist computing. *Progress in Computing, Analytics and Networking*. New York, NY: Springer; 2018:733-742.
40. Aazam M, St-Hilaire M, Lung CH, Lambadaris I, Huh EN. IoT resource estimation challenges and modeling in fog. *Fog Computing in the Internet of Things*. Cham: Springer; 2018:17-31.
41. Skarlat O, Schulte S, Borkowski M, Leitner P. Resource provisioning for IoT services in the fog. Paper presented at: Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA); 2016:32-39; IEEE.
42. Bello O, Zeadally S. Intelligent device-to-device communication in the internet of things. *IEEE Syst J*. 2014;10(3):1172-1182.
43. Ansari RI, Chrysostomou C, Hassan SA, et al. 5G D2D networks: techniques, challenges, and future prospects. *IEEE Syst J*. 2017;12(4):3970-3984.
44. Wu X, Tavildar S, Shakkottai S, et al. FlashLinQ: a synchronous distributed scheduler for peer-to-peer ad hoc networks. *IEEE/ACM Trans Netw (ToN)*. 2013;21(4):1215-1228.
45. Ngu AH, Gutierrez M, Metsis V, Nepal S, Sheng QZ. IoT middleware: a survey on issues and enabling technologies. *IEEE Internet Things J*. 2016;4(1):1-20.
46. Shahraki A, Rafsanjani MK, Saeid AB. A new approach for energy and delay trade-off intra-clustering routing in WSNs. *Comput Math Appl*. 2011;62(4):1670-1676.
47. Shahraki A. Datasets related to the paper: a comparative node evaluation model for highly heterogeneous massive-scale IoT-Mist networks. OPNET MODELER; 2019. https://doi.org/10.17632/fz7d3h92b4.2.
48. Chen T, Guestrin C. Xgboost: a scalable tree boosting system. Paper presented at: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016:785-794; ACM.
49. Geitle M, Olsson R. A new baseline for automated hyper-parameter optimization. In: Nicosia G, Pardalos P, Umeton R, Giuffrida G, Sciacca V, eds. *Machine Learning, Optimization, and Data Science*. Cham: Springer International Publishing; 2019:521-530.
50. Shahraki A, Geitle M, Haugen Ø. A distributed fog node assessment model by using fuzzy rules learned by XGBoost. Paper presented at: Proceedings of the 2019 4th International Conference on Smart and Sustainable Technologies (SpliTech); 2019:1-6; IEEE.

How to cite this article: Shahraki A, Geitle M, Haugen Ø. A comparative node evaluation model for highly heterogeneous massive-scale Internet of Things-Mist networks. *Trans Emerging Tel Tech*. 2020;1–28. https://doi.org/10.1002/ett.3924