Wireless Personal Communications (2022) 125:1973–2002
https://doi.org/10.1007/s11277-022-09643-9

MMF Clustering: A On-demand One-hop Cluster Management in MANET Services Executing Perspective

Reza Sookhtsaraei[1] · **Mohsen Nejadkheirallah**[2] · **Mohammad Saber Iraji**[1,3]

Accepted: 7 February 2022 / Published online: 13 March 2022

Abstract

Mobile Ad-hoc Network (MANET) consists of a group of mobile nodes that communicate without any infrastructure. The dynamic nature and intrinsic complexity of MANET have made it a network with high topological variability. It is highly desirable to find methods that bring this complexity under control quickly. Controlling this complexity makes communication between nodes more durable, resource utilization more efficient, and the quality of required services from the environment higher. Since clustering is one of the most common methods for overcoming flat structures with many nodes, researchers have always looked for practical algorithms for clustering in MANET. Therefore, in this research, an attempt has been made to provide a method for clustering nodes in this environment to make the clusters more stable. To achieve more stable clusters, parameters for header selection are considered that reduce the need to change the header in each cluster. Also, in addition to creating new clusters if necessary, by constantly monitoring the performance of existing clusters, as far as possible, these clusters are reorganized and reconfigured to have more stable clusters in the environment. The simulation results show that creating more stable clusters in MANET leads to more efficient node resources and higher service quality than existing methods.

Keywords MANET · Cluster · Resources management · QoS

✉ Mohammad Saber Iraji
iraji.ms@gmail.com

Reza Sookhtsaraei
reza.sookhtsaraei@gmail.com

Mohsen Nejadkheirallah
m.nejad@mohaddes.ac.ir

1 Department of Computer Engineering and Information Technology, Payame Noor University, Tehran, Iran

2 Department of Computer Engineering, Allameh Mohaddes Nouri University, Noor, Iran

3 Department of Computer Engineering, University of Tabriz, Tabriz, Iran

1 Introduction

MANET is one of the post-PC era phenomena. The post-PC era was brought up by leading companies such as IBM [1]. Some applications such as battlefields, law enforcement, mobile conferencing, disaster relief, and emergency rescue cannot be used by fixed and pre-built infrastructure such as access points on wireless networks or routers on wired networks for communications. Mobile networks are used to establish communication in these applications. In recent years, mobile networks, especially MANETs, have attracted a great deal of attention as the rapid advancement of wireless communication technology and its popular technology, i.e., smartphones. MANET is a wireless self-organizing network whose mobile nodes are connected without any fixed infrastructure [2–9]. Many standard technologies such as Ultra-Wideband (UWB), IEEE 802.15.3 [10] (Wireless PAN), IEEE 802.11 [11] (Wi-Fi), Bluetooth [12] support MANET. In such an environment, accessing suitable resources and running services effectively due to limited communication range and high mobility of nodes is very challenging [13]. Various strategies have been proposed to find suitable resources and implement the services in MANET, which can be categorized into four types of blind [14, 16], precise [17, 18], heuristic [6, 19, 20], and other types [5, 21–23].

When the size of a network grows, regardless of the used routing method, a hierarchical network structure will perform much better than a flat structure [24–26]. When the nodes are mobile, and the network structure is flat, the network will become very non-scalable. The computational complexity of finding a node in a flat network is O (n^2) [27]. Therefore, a hierarchical structure is necessary to achieve the required minimum performance in a high-scale MANET [28]. Cluster structure is a common implementation in hierarchical architecture. But in the subject under discussion, cluster formation and header selection appropriately is an NP-hard problem [29]. Since clustering is one of the most common hierarchical methods, researchers have always sought effective clustering algorithms in MANET. Clustering techniques can group mobile nodes based on specific features and create a more scalable environment. In this regard, to eliminate the shortcomings of clustering methods in MANET, a method for clustering nodes in this environment will be provided, which aims to manage resources better and provide the maximum quality of services requested by MANET.

The contributions addressed in this study are:

A- Provides a comprehensive and far from a complex solution for managing clusters in MANET, including creating, maintaining, troubleshooting, and stopping them from increasing the efficiency of the environment.

B- Increases the stability of the clusters in the environment by constantly monitoring the requirements of the clusters.

C- Reorganizes clusters in the environment by considering effective parameters to reduce the reorganization frequency in MANET as much as possible.

This research is organized so that in Sect. 2, the clustering algorithms used in MANET are reviewed. Section 3 presents the basic cluster management operations in MANET. In Sect. 4, solutions for more clusters stability in MANET are presented. In Sect. 5, the MANET management middleware and its components are introduced. In Sect. 6, the proposed method is compared and analyzed with one of the efficient one-hop clustering methods in the desired environment. Finally, the present study ends in Sect. 7, with the conclusion and expression of further research in the future.

2 Related Works

A variety of Clustering approaches has been studied in distributed environment for decades. Clustering and hierarchical patterns can be subdivided into different categories based on the used parameters and methods. Here are some of the most important recent works.

Ergenç et al. [30] presented a Dependability-based Clustering Algorithm (DCA). This algorithm has two steps that are executed sequentially. Initially, initialization is done to form the cluster. Then, by sending messages, some important parameters that affect the reliability of the clusters and nodes are continually monitored and by an equation, the reliability score of each cluster and each node is calculated. In the next step, which is the maintenance phase, tasks such as evaluating the neighbor's information, selecting the cluster head, and selecting the cluster as an infinite procedure are done until the network exists. DCA is a weighted clustering algorithm in which a combination of parameters is considered.

Clustering methods are studied for different purposes as well as in extensive research fields. Vehicle Ad-hoc Network (VANET) is an important topic for using clustering algorithms that impose additional constraints such as high-level mobility. Therefore, Yang et al. [31] have used the remaining path time, which is only a measure of the total neighborhood time, as the only clustering criterion in VANETs. Although path information plays an important role in the formation of more stable clusters, this approach relies heavily on the predetermined paths of nodes and cannot use nodes with random paths in cluster formation. The algorithm uses a parameter to form clusters.

Han et al. [32] also used a clustering method for the Internet of Medical things. Since one of the main concerns of modern medical systems, especially Internet of Things (IoT) based medical systems, is the conservation of medical devices to extend the lifespan of the healthcare system, clustering will play an important role in this energy saving. The method proposed by them consists of two stages: header selection and cluster formation. The basis of header selection is an equation that selects a node that has the right amount of resource capacity parameters, queue capacity, distance to other nodes, and energy. Once the header is selected, a broad message is sent to the rest of the nodes and they are asked to provide their received medical data. The research employed a clustering algorithm focusing on energy efficiency.

Chaudhry et al. [33], proposed a method for controlling network scalability using clustering for MANET. This method uses the modified Gabriel graph algorithm and determines the parameters such as transmission energy, amount of energy consumed per node and network delay, the optimal amount of power required by the nodes to exchange with other nodes, and the longer shelf life of the clusters. The clustering algorithm used by them can be classified as cluster weight-based clustering or a mobile node-based clustering algorithm.

Nabar et al. [34], proposed a Gaussian Markov distributed clustering approach with the node-based diffusion policy in MANET. Its name is GMM-APD algorithm. The algorithm uses heuristics for greedy clustering. GMM-APD is proposed for overcoming the disadvantages of greedy methods, for example the local optimal concept. In this algorithm, the motion pattern of each node is shown by the Gauss-Markov distribution and the mobility of nodes is used as a criterion to evaluate the proximity of the nodes to each other. The GMM-APD can be further described as a node mobility-based clustering approach.

Bhushan et al. [35], have proposed an algorithm named fuzzy attribute-based joint integrated scheduling and tree formation (FAJIT) technique for tree formation in a Wireless sensor network (WSN). FAJIT mainly focuses on addressing the parent node selection problem in the heterogeneous network for aggregating different types of data packets to improve energy efficiency. The selection of parent nodes is performed based on the candidate nodes with the minimum number of dynamic neighbors. Fuzzy logic is applied in the case of an equal number of dynamic neighbors. In the proposed technique, fuzzy logic is first applied to WSN, and then min–max normalization is used to retrieve normalized weights (membership values) for the given edges of the graph. This membership value is used to denote the degree to which an element belongs to a set. Therefore, the node with the minimum sum of all weights is considered as the parent node.

Rao et al. [36], have provided a hierarchical routing protocol with awareness of service quality, especially with the aim of energy efficiency for MANET. The name of this algorithm is KF-MAC, which uses the mean K-based clustering method based on MAC-based routing to determine the cluster heads. Initially, the mean K clustering technique is used to cluster nodes. Then, based on the firefly optimization algorithm, the nodes are clustered, sorted and optimized to select the most appropriate nodes for the cluster heads. KF-MAC uses network parameters such as bandwidth, delay, bitrate and jitter to provide quality of services.

By Ahmad et al. [37], at the beginning of the work, cluster formation is performed. Then, an algorithm based on bee algorithms, genetics and tabu search is presented for the IoT. In this algorithm, each chromosome shows a possible structure for the cluster, and the suitability of each candidate cluster is evaluated by its compatibility and load balance. In fact, by integrating the features provided by the bee and genetic algorithms, this method utilizes populations that generate network dynamics that represent different cluster structures, and produce high-quality solutions. Obviously, the clustering method is a nature-based method with the aim of optimizing as many parameters as possible.

Since the discussion of resource management in MANET is very prominent in the present study, it is necessary to pay attention to the researches that have considered the use of resources in this field. Therefore, by Meng et al. [38], efficient resource searching in MANET is considered. Two intrinsic issues related to MANET, the limited range of wireless communications as well as the high mobility of nodes, are discussed in this paper and a method called tieSearch is proposed to overcome them. tieSearch uses the ant colony algorithm, where each pheromone contains information about the neighbors' availability pattern each node and the resources in them. Actually, this paper does not explicitly address the problem of clustering, but each node together with its neighbors selected by pheromones constitutes a cluster implicitly. To achieve the neighbors' availability pattern of each node and their resources, in tieSearch, a period, such as a day, is divided into several smaller intervals and the pattern of neighbors' movements and transactions they perform across Those intervals, are checked.

Chithaluru et al. [39] have adopted the idea of IoT for constructing a green wireless sensor network (WSN) for improving sensor based communication in future smart cities. To achieve green IoT implementation, it is important to take necessary measures to prevent energy depletion and promote energy efficiency techniques. Clustering can extend the lifetime of such networks and its efficiency depends on the selection of quality clustering schemes. They have proposed I-AREOR to balance the energy consumption for maximizing the network lifetime based on regional density, relative distance, and residual energy.

Srivastava et al. [40], have focused on the advancement of a cloud administration's provisioning structure by building up a dynamic load-balancer for the cloud. For load balancing, gossip protocol has been used for inter/intra-cluster gossip. For inter-cluster gossip, the load is balanced among the leaders of every cluster. The proposed protocol uses the inter-cloud resource management, where a leader is selected from the cloud that interacts to other cloud and decides on virtual machine (VM) migration.

In short, if you want to compare this work with recent works, there are a few important points to consider: In this work, as an essential solution to overcome the complexity of MANET, the cluster management process has been dealt with extensively so that a researcher can read most aspects of cluster management in this environment by reading it. In this paper, a combination of main environmental parameters derived from the dynamic nature of MANET is investigated so that a proper evaluation of the environment cannot be achieved without considering each of them. According to our information, the current research is the first in MANET to address this environment from the service execution perspective. Last but not least, in this work, due to the inherent complexities of MANET, it is tried to use effortless greedy methods to deal with problems. We always think that dealing with problems using sophisticated methods in complex environments makes the situation more complicated.

3 Basic Cluster Management Operations in MANET

As mentioned in the previous sections, this work aims to provide a fairly complete and simple way to manage clusters in MANET. It has been said before that clustering is one of the most effective ways to overcome the high dynamics of this environment. Clustering in MANET means dividing geographic environments into smaller areas and forming smaller virtual groups so that these small virtual groups cover the whole environment. Splitting nodes into several clusters has the following advantages:

(a) It makes the system scalable.
(b) It Minimizes lengthy and costly message-based communications between nodes.
(c) It Increases accessibility resulting in increased service delivery locally.

There are generally three types of nodes in each cluster:

(a) Cluster head: depending on the specific parameters or the determining factors, a node is selected as the cluster head. The cluster head is responsible for coordinating communications between other cluster nodes and maintaining the cluster.
(b) Gateway node: it is a node that manages the communication of a cluster with adjacent clusters. This node communicates between two or more clusters. However, it is possible that for simplifying communication processes, a cluster may have a more significant number of gateway nodes. In some research, including the present study, the gateway nodes are ignored, and the node header takes the task.
(c) Normal node: any node in a cluster that is not a header or a gateway is normal.
(d) The following parameters are examined to form clusters, determine the header and other cluster members and rank them.

3.1 Investigating the Determinant Parameters of Each Node

In this research, prioritizing the nodes to achieve the goals of the proposed method is a necessary process. Therefore, it is necessary to consider the parameters that put a node in higher priority than other nodes. The parameters that are evaluated to rank each node i, include the total processor capacity and the total node memory whose units are $MIPS_i$ and MB_i, respectively. The next important parameter is the average battery discharge time in the previous few periods (Eq. 1) stored in the device's history data.

$$D_i = \frac{\sum (Sod - Eod)}{n} \tag{1}$$

where Sod is the time to start using the battery in the charged state, and Eod is the time to reach the last p percent of battery charge. n and D, are respectively the mean discharge time of the device i (seconds) and the number of battery discharge periods to calculate the mean discharge time of the battery. Other parameters include the amount of assigned CPU and assigned memory at the current time of node i, which is represented in $mips_i$ and mb_i, respectively.

Another parameter is the trust level of a node based on the performance history of that node. (Eq. 2)

$$\varphi_i = \varphi_i + \alpha c \tag{2}$$

In Eq. (2), φ_i is a value represents the trust degree of node i in MANET. This parameter is equal to 1 at the beginning of the arrival of each device to MANET. But *c* will vary based on the performance of each device and the role that device plays in MANET. If device i in the cluster is in the header role, c is equal to ε_1 and if device i is the normal member of the cluster, value c will be ε_2 ($\varepsilon_1 > \varepsilon_2$). If the device i performs its task correctly during each period, the value of α will be + 1, and if it fails or performs its task incorrectly, α will assign a value of −1. Accordingly, the value of φ_i will be dynamically adjusted according to the role of each node. Algorithm 1 shows the trust calculation of a node.

Algorithm 1: TrustOf (NodeType type, TrustValue φ, boolean isFault)

```
if (Node.type = = header)
    if (isfault)
        $\varphi = \varphi - \varepsilon_1$
    else
        $\varphi = \varphi + \varepsilon_1$
else
    if (isfault)
        $\varphi = \varphi - \varepsilon_2$
    else
        $\varphi = \varphi + \varepsilon_2$
end.
```

The history data for each device is stored in a middleware called MMF (Sect. 5) that only the MANET infrastructure will have access to it. In other words, the data needed to validate MANET will not be manipulated by unauthorized persons.

The most important member of any cluster is the cluster header because cluster stability is highly dependent on header stability. To achieve this goal, among the available nodes, a node should be selected as the header of each MANET cluster, which is more resistant to significant changes in this environment and can create a more durable cluster by managing the workloads sent to it. Therefore, the best criteria for selecting the header should be considered to choose the most powerful node in terms of resource durability for this role among the nodes in each cluster. The result of the study of these criteria has led to Eq. (3). Equation 3 can be used to select the device with the highest value as the appropriate node in the header:

$$Max\left(k_i(D_i\omega_i + \varphi_i + \left(MIPS_i - mips_i\right) + \left(MB_i - mb_i\right))/\Psi_1 T_{sr}\right)\forall i \in n \tag{3}$$

where k_i, is the number of specific resources available in node i other than processor and memory, ω_i is battery capacity percentage of node i, n is equal with the total number of devices examined for clustering, T_{sr} is the average waiting time for specific resources in node i and Ψ_1 According to the following equations, is the average of movement difference between node i and the average of movement of the whole cluster is:

$$\Psi_1 = \left(\frac{\sum_{i=1}^{n} \mathrm{M}_{\mathrm{iave}}}{\mathrm{n}} - \mathrm{M}_{\mathrm{ave}}\right) \tag{4}$$

$\mathrm{M}_{\mathrm{ave}}$ is also calculated as follows:

$$M_{ave} = \frac{1}{\mathcal{T}}\sum_{t=1}^{\mathcal{T}} \sqrt{(x(t) - x(t-1))^2 + (y(t) - y(t-1))^2} \tag{5}$$

Equation (5) shows the average of a device movement. Using Eqs. (4) and (5), a node that has been closer to the other members of that cluster during its lifetime in a cluster will be of greater importance in Eq. (3).

3.2 Forming a Cluster or Joining an Existing Cluster

When a node has a demand for extra resources or a specific resource that does not exist in that node, it broadcasts the demand. Broadcasting this demand is the first step to join that node to one of the existing clusters or to form a new cluster. In the current study, one-hop clusters are examined. One-hop clusters are clusters where the nodes are located near the header and have access to it directly. The following situations may now occur to this message broadcasting:

a. At least one header exists within the range of the message sending device.
b. Even a header doesn't exist within the range of the message sending device.

The service requesting device broadcasts a message in the following format:

$$\left\langle ID.\mathcal{R}_{req}.\mathcal{R}_{ava}.M_{ave}\right\rangle \tag{6}$$

where the ID indicates the unique identifier of the device in MANET, $\mathcal{R}_{req}$ refers to needed resources to run the service. $\mathcal{R}_{ava}$ and M_{ave} also show the resources on the device to share in MANET and the average movement of the device, respectively.

4 At Least One Header Exists Within the Range of the Message Sending Device

As shown in Fig. 1, the headers that exist in the message domain sent by the service requesting device, receive the message and then evaluate it by Eqs. (4), (8), (9), and (10). Then the headers declare their suitability for joining the applicant device to them by a new message of type (7):

$$\langle ID.\Psi \rangle \tag{7}$$

The ID is the unique cluster number in MANET and Ψ indicates cluster suitability for joining the requesting node to that cluster. The calculation method of Ψ is as follows.

$$\Psi = \frac{\left(\Psi_1 + \Psi_2\right)}{\varphi} \tag{8}$$

$$\Psi_2 = \sum_{i=1}^{m} \left(\left(\left(R_{cluste-ava}(i) \right) - R_{req}(i) \right) + \left(\left| R_{ave}(i) - \left(R_{cluster-req}(i) \right) \right| \right) \right) \tag{9}$$

Ψ_1 calculates the average movement difference between the applicant node and the average movement of the cluster's members. Ψ_2 identifies a cluster that needs more to the applicant

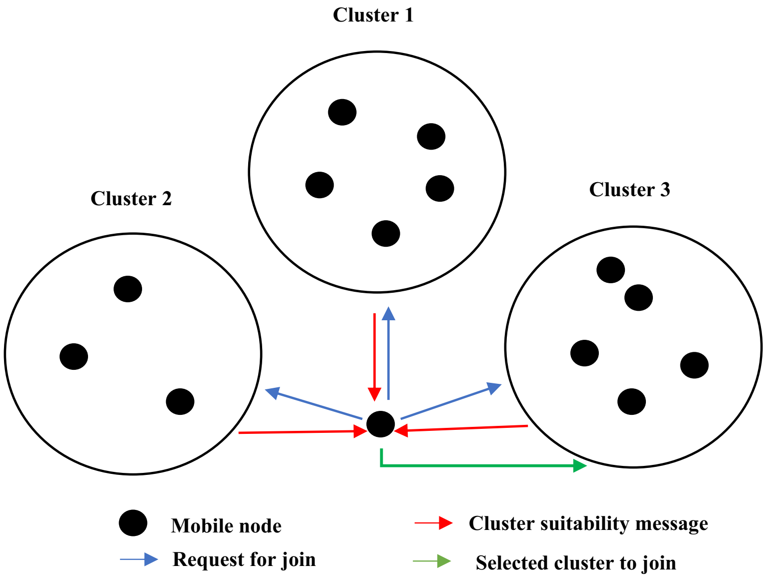


Fig. 1 Steps to join a node to a cluster in MANET\

node's surplus resources and also provides needed resources of that node (Of course, the first sentence of Eq. (9) should not be negative). φ and m represent a header trust and the types of available resources in MANET, respectively. In order to save more in resources, the node joins a cluster that minimizes Ψ. in other words:

$$\mathrm{Min}\left(\Psi_{\mathrm{i}}\right)\forall \mathrm{i} \in \mathrm{ClustersSet} \tag{10}$$

5 Even a Header Doesn't Exist Within the Range of the Message Sending Device

The requesting node broadcasts a message containing its identification number and needed resources. If they need resources of that node, devices around the applicant device send a positive message along with their movement average and their coordinates to that device when they receive and process the message. The service requesting node does the job of accepting the surrounding nodes as long as it still needs resources. For this purpose, after receiving the message from the surrounding nodes, it arranges them in ascending order according to Eq. (11) and selects a number of them as needed from the beginning of the sorted list. Now, the service requesting node and others accepting by it form a cluster together. According to Eq. (3), the nodes are rated in this cluster, and the node with the highest score accepts the header role for the cluster.

$$\frac{\left|M_{ave} - M_i\right| + \sqrt{(x - x(i))^2 + (y - y(i))^2}}{\varphi_i} \tag{11}$$

where M_{ave} is the movement average of service request node, M_i, is the movement average of a service demand response node, $< x.y >$ is service request node coordinates, $< x_i.y_i >$ node i coordinates and φ_i is the trust of node i in MANET. Algorithm 2 shows the method of joining a node to an existing cluster. The algorithm for forming a cluster is presented in more detail in Sects. 4.1 (Algorithm 3,4), as the need arises.

Algorithm 2: JoinCluster (Node n, MessageCollection s)

```
$c = +\infty$
for each cluster $c_i$ that it's response is in s
        $\Psi_{c_i}$= n calculates equation (8)
        if (c > $\Psi_{c_i}$)
            index = i
            $c = \Psi_{c_i}$
$n\ joins\ c_i$
sc= $c_i$ finds nearest $sc_j$ to n  // sc is sub-Cluster that in the following sections will explain.
$n \in sc$
```

5.1 Adding a New Node to MANET

The node that enters MANET has one of two following purposes: first, that node needs a service that, in this case, will be treated like Sect. 3–2. Second, its purpose is to cooperate with MANET. For the latter purpose, the new node, broadcasts a message that contains its ID and resources. The headers within the new node range, if needed to the resources provided by that node, send a positive message containing the header coordinates and the required amount of each resource in that cluster to the new node. After evaluating the respondent clusters, the new node selects the cluster that, according to formula (12), needs more resources from that node and is closer to that node to join.

$$\sum_{i=1}^{m} \left(C_{j-req}(i) - R_{ave}(i)\right) + \sqrt{(x - x(i))^2 + (y - y(i))^2} \tag{12}$$

where $C_{j-req}(i)$ is the value of the cluster j requirement for resource i and the remaining symbols used in this equation are presented in the preceding sections.

5.2 Removing a Node From MANET

Removing a node from a cluster may be for one of the following reasons:

(1) The lifetime of the cluster task is over.
(2) The node with its discretion tends to leave MANET.
(3) The cluster no longer needs a node.
(4) Due to the node's distance from the header, it is not appropriate to keep it in the current cluster.
(5) Due to a fault, the node cannot be accompanied by the cluster.

In case #1, if recurring reconfigurations occur for a cluster, maintenance of that cluster will no longer be useful for MANET, and hence the end of the task of those nodes in that cluster will be notified to the nodes.

In case #2, each node may abandon its cluster for various reasons (if it is not running a service). For example, one node may find that it is more efficient in another cluster, that is, it has a higher rank in other clusters. It will, therefore, freely drop its cluster and act as a new node. But otherwise, if a node leaves a cluster during the execution of the service, if the node itself is responsible for performing all or part of that service, the node's trust will be reduced by Eq. (2).

In case #3, if a node is not used to run part or all of those services after several services are assigned to one cluster, that node in the cluster is no longer needed, and the corresponding header declares the message of that node's detachment from the current cluster.

In case #4, it means that a node may no longer receive messages directly from the header because of the distance from its cluster header, which would indicate removing that node from the current cluster.

In case #5, the node will stop working due to a fault. In this work, due to the breadth of the discussion, fault management is not raised in MANET.

6 More Cluster Stability: Reorganize or Reconfigure Clusters in MANET

For various reasons, such as header energy depletion, high load accumulation on it, the distance of some nodes to the header and other cluster nodes, cluster load imbalance, etc., may decrease cluster performance. One important way to prevent a cluster from reducing performance is to reconfigure it. In order to measure performance, the header node must have continuous monitoring of all its cluster nodes. To perform this type of monitoring, transmitting messages between all cluster nodes with its header is necessary. Sending messages should also follow a specific pattern, which can be based on an event occurrence or use a regular periodic pattern.

6.1 Prerequisites for Cluster Reconfiguration

As mentioned, at the beginning of a cluster formation, a list based on Eq. (3) is formed in descending order by the cluster generator node. This list contains an ID for each node of the cluster. From now on, this list is called the Cluster Neighborhood List (CNL). The nodes at the top of this list have higher computational and security credentials and better conditions for performing services assigned to the cluster. Therefore, the node at the top of this list is selected as the cluster header. It can now be reduced the regulatory and management complexity in the cluster using this list and by registering the location of each node at the beginning of the cluster formation in it. For this purpose, starting at the end of the list and based on Eq. (13), the nearest m nodes to each node are calculated, and their location is inserted into the CNL entry of that node.

$$\sqrt{(x-x(i))^2+(y-y(i))^2} \tag{13}$$

After calculating m neighbors for each node from the end of CNL, is no longer performed this calculation for m closest neighbors found. This reduces the repetition of finding neighbors in the cluster. Now the question may arise in the reader's mind as to why finding the nearest neighbors begins at the CNL end nodes! The answer is that, according to Eq. (3), the chances of nodes of top of the CNL are greater for running the services delegated to the cluster because of the higher computational and security validity than the end nodes. As a result, the workload in the early nodes will be greater than the CNL end nodes. Therefore, in order to distribute the workload evenly across cluster nodes, is delegated the task of managing the exchanged messages within a cluster to the less-loaded nodes at the bottom of the CNL. To the trust value of each node responsible for the message exchange in the cluster, upon successful completion of the tasks assigned to it, will be added according to Eq. (2) so that it can be placed in later transitions closer to the beginning of CNL. The header selection procedure, of course, is an exception to this because of the sensitive tasks it performs.

For easier access to the nodes of each cluster, is inserted m nearest neighbor of each node along with their location in the CNL. CNL with the two closest neighbors for each node is shown in Fig. 2.

ID_1		ID_2		ID_3			ID_n	
3	(x_3,y_3)			1	(x_1,y_1)		1	(x_1,y_1)
n	(x_n,y_n)			n	(x_n,y_n)		3	(x_3,y_3)

Fig. 2 Neighborhood list of a cluster with two neighbors for each node

Algorithm 3: CreateCluster (Node n, MessageCollection s)

```
n sorts nodes in s base on equation (11) by decreasing order
while (needs of n have not been resolved)
        n selects a node from the beginning of the arranged list
for each node $n_j$ in s
    CNL($n_j$).add(Calculate equation (3))
Sort CNL by decreasing order
for k = CNL.lenght() to 1
    if (!CNL.mNearest(k))
        CNL(k).mNearest=find m nearest nodes
$n_{header}$= CNL(1)
n sends CNL to $n_{header}$
$n_{header}$ identifies subheaders $HSC_j$ and sends the relevant information to them
$n_{header}$ waits for all subheaders response each $\Delta t_1$ time
```

Consider each node and its neighbors as a sub-cluster for the current cluster, and we denote it by SC_{ij}, representing the subgroup j of cluster i. After the CNL is formed and the header and the nearest neighbors are identified, the CNL is sent to the header. The steps of forming a cluster are shown in Fig. 3 and its pseudo-code by Algorithm 3. After receiving the CNL and checking the nodes from its end, the main header sends information about each sub-cluster to the header of that sub-cluster. From now on, each sub-cluster must monitor itself. Cluster header declares to their sub-clusters that I am the primary header, and only the headers of each sub-cluster must send their sub-cluster information to the primary header after a specified period of time (Δt_1). Let us denote the header of sub-cluster j of cluster i with HSC_{ij}. This header is one of the same nodes examined from the CNL ends. HSC_{ij} sends messages to its designated neighbors in CNL after receiving its sub-cluster information. HSC_{ij} introduces itself in this message and asks them to report their status and locations to it within a specified period of time (Δt_2).

Algorithm 4: ToJoinOrCreateCluster ()

```
$n_i$ broadcasts message $< ID.\mathcal{R}_{res}.\mathcal{R}_{ava}.M_{ave} >$
$S_{Header}$= $n_i$ collects responses from headers that located around itself in a given time.
$S_{Normal}$= $n_i$ collects responses from normal nodes that located around itself in a given time.
if ($S_{Header} \neq \emptyset$ )
    JoinCluster ($n_i$ , $S_{Header}$)
else
    CreateCluster($n_i$ , $S_{Normal}$)
end.
```

After receiving this message, other sub-cluster nodes send the confirmation of receipt to their sub-header. After each $\left(\Delta t_2\right).(\Delta t_2 < (\frac{\Delta t_1}{2}))$, the status messages of each sub-cluster node is sent to the corresponding header and a confirmation message is received. In each message exchange, the positional information of the nodes is sent to each other so that the sub-header

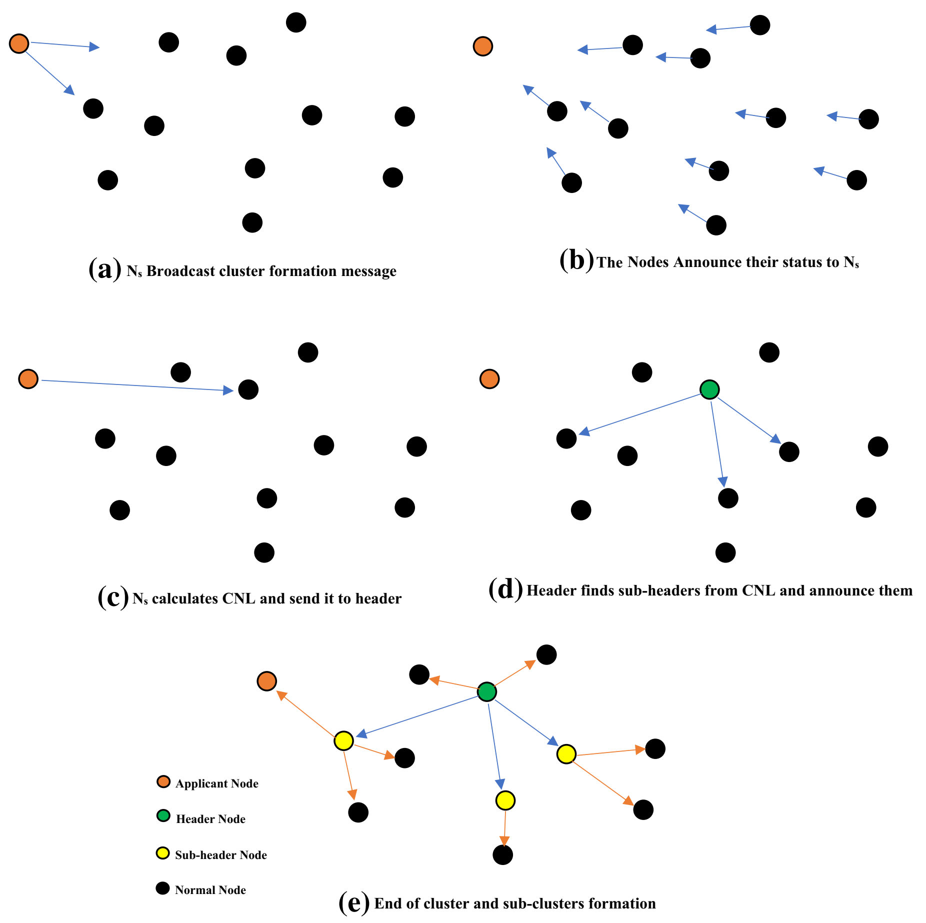

(a) Nₛ Broadcast cluster formation message
(b) The Nodes Announce their status to Nₛ
(c) Nₛ calculates CNL and send it to header
(d) Header finds sub-headers from CNL and announce them
Applicant Node
Header Node
Sub-header Node
Normal Node
(e) End of cluster and sub-clusters formation

Fig. 3 The steps of forming a new cluster with its sub-clusters along with two neighbors

constantly tracks their position. The critical task of the header in each sub-cluster is to calculate Eq. (3) for all the nodes of the target sub-cluster. Algorithm 4 is the starting point for joining a node to MANET or forming a new cluster in it.

6.2 When is the Reconfiguration Done?

One of the most important criteria that can determine the reconfigure time of a cluster is its performance criterion. Here, the meaning of performance is equal to a cluster's success in providing the requested services. For this purpose, the following are presented to measure the performance of each cluster:

6.2.1 The Number of Services Performed by the Cluster Relative to the Total of Requested Services by MANET in a Given Work Period:

In addition to intra-cluster communications, clusters also interact with other clusters in the environment. The head and gate nodes of each cluster are bridges for inter-cluster communication. In this study, inter-cluster communications are performed by headers and gateway nodes are ignored. It is also assumed that inter-cluster communications are used to exchange messages containing the headers' geographical location, the number of nodes in each cluster, the number of requested services from each cluster, the number of performed and violated services by that cluster. From the data in inter-cluster communication messages, the time to reconfigure a cluster can be determined. Based on the inter-cluster message exchange, one can calculate the ratio of the number of performed services by the cluster i to the total requested services by MANET according to Eq. (14), in which PS_i is the number of performed services by the cluster i, RS_j is the number of requested services from the cluster j and C is the total number of clusters available in MANET:

$$\alpha = \frac{PS_i}{\sum_{j=1}^{C} RS_j} \tag{14}$$

6.2.2 Number of Violated Services to Total Accepted Services by the Cluster

Another indicator that directly impacts the performance of a cluster is the number of violated services by that cluster. The previous sections explained how to send a request to MANET and select the appropriate cluster to service that request. When the accepted service by a cluster is violated, MANET reduces the credibility of that cluster by reducing its trust in header according to Eq. (2), and thus reduces the rate of outsourcing more services to that cluster. This is done by Eqs. (8) and (11). On the other hand, maintaining clusters that run fewer services is not economical for MANET. Therefore, to identify these types of clusters, Eq. (15) is presented which VS is the number of violated services and AS is the accepted service in a cluster:

$$\beta = \frac{VS}{\sum_{i=1} AS_i} \tag{15}$$

6.2.3 High Scattering of Cluster Members

Another major cause of service violation or disruption of mobile nodes' activity is the long distance between cluster members. Consider the situation in which nodes in a cluster are involved in running a service and require intra-cluster message exchange to continue running the service. But because the nodes are so far apart, messaging to update data is slow or not done. For this purpose, it is necessary to consider the distance between the nodes. This is done by Eq. (16), where subH is the number of sub-headers in a cluster and m the number of normal nodes per sub-headers:

$$\gamma = \frac{1}{subH} \sum_{j=1}^{subH} \frac{1}{m_j + 1} \left(\sum_{k=1}^{m_j} \sqrt{\left(x_j - x_k\right)^2 - \left(y_j - y_k\right)^2} \right) \mathit{if\ node}_k \in C_j \mathit{and\ not\ exist\ in\ j\ domain\ then}\ (x_k . y_k) = \infty \tag{16}$$

Now that necessary values to calculate the performance of a cluster exists, the best time to reconfigure the cluster should be determined. Whenever a service is assigned to a cluster and the service is denied or violated by the cluster header, a "Reconfiguration" message is sent to all the headers in MANET by that cluster. From now on, this cluster is called C_{Config}. All headers (no sub-headers) send the calculated value of Eq. (17) to C_{Config}. C_{Config} sorts all the answers after receiving them in ascending order. Based on where C_{Config} is placed in the sorted list, it determines whether that cluster will reorganize/reconfigure or not.

$$F_{re-config} = \frac{1 + \alpha(\beta + \gamma)}{\alpha} \tag{17}$$

6.3 What Happens in Reconfiguration for a Cluster?

Service rejection and service violation are two important criteria in determining the performance of a cluster. To this end, and based on the criteria outlined in the preceding section, the following can be considered as reasons for rejecting or violating the Service:

-Not enough resources to continue the service.

-Node failure while running the service.

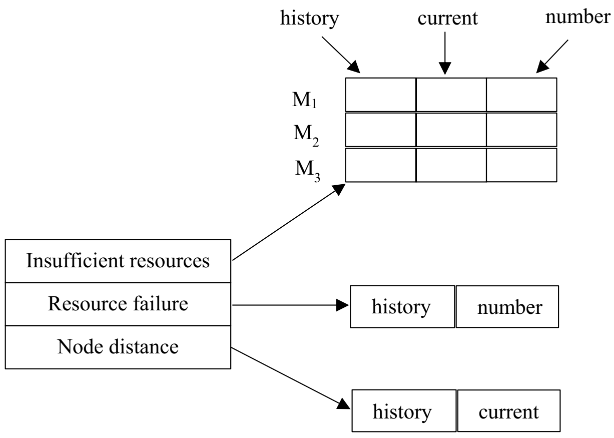
-Increasing the distance of cooperating nodes in service execution from one another.

When rejecting or violating a service, the reasons for this event are stored in the MMF reconfiguration section of the cluster head. As previously mentioned, each cluster decides to reconfigure itself after each message exchange with the other clusters in MANET and based on the calculated value of Eq. (17). Here's what a cluster does to execute its reconfiguration process. The C_{Config} header first looks at the ReConfig section of the MMF, exploring the reasons for the performance loss, which is one of the three factors mentioned above.

In the header's ReConfig section of the MMF, a data structure called the Cluster Events List (CEL), as shown in Fig. 4, is used. In this data structure, the first entry, which indicates the scarcity of resources, refers to an array of resources with three fields in each row.

The history field shows the average requirement from a resource in past events for which requests to the cluster was rejected or violated for that reason. The current field represents the aggregate of a needed resource in the current configuration of the cluster that rejects or

Fig. 4 CEL with three resource



violates a service, and the number field represents the total number of rejected or violated services in the current configuration of a cluster. The “Resource failure” section shows the average number of nodes crashes in previous periods as well as the number of crashes in the current configuration. “Node distance” entry also refers to the average distance of nodes in sub-clusters in previous and current periods. Now, based on Eqs. (18) to (22), the importance of each CEL event is calculated and, after normalizing them in [0,1], are sorted into a separate list in descending order.

$$M_1 = (1 - \partial) history_{M_1} + \partial\left(\frac{current_{M_1}}{number_{M_1}}\right) \tag{18}$$

$$M_2 = (1 - \partial) history_{M_2} + \partial\left(\frac{current_{M_2}}{number_{M_2}}\right) \tag{19}$$

$$M_3 = (1 - \partial) history_{M_3} + \partial\left(\frac{current_{M_3}}{number_{M_3}}\right) \tag{20}$$

$$RF = \frac{history_{RF} + number_{RF}}{2} \tag{21}$$

$$ND = \frac{historyAVE_{ND} + currentAVE_{ND}}{2} \tag{22}$$

RF generated from “Resource failure” and ND also generated from “Node Distance” words. ∂ Also considered as the coefficient of significance of the current value in the final value of an event and its value is between [0,1].

For each service request sent to a cluster, three modes of violation, rejection, or successful execution may occur. In case of rejection, for each service, the current average requirement of M_i resource relative to the average history of that resource is checked. In other words, if $\mathrm{history}_{\mathrm{M_i}} \leq \left(\frac{\mathrm{current}_{\mathrm{M_i}}}{\mathrm{number}_{\mathrm{M_i}}}\right)$, it will be the start point of changing the current configuration of the cluster. But conditions are different for service violations. In service rejection, the cluster has made no commitment to perform the service, but in service violation, the cluster has accepted the service but has failed to fulfill its obligations.

Algorithm 5: PerformanceChecking ()

```
ICF=false      //InterCluster Communication Flag
prrArray=Null   //Performance Reduction Reason
for each $M_i$
    if $history_{M_i} \leq \left(\frac{current_{M_i}}{number_{M_i}}\right)$
        ICF=true
        prrArray.add($M_i$)
end for
if (!ICF)
    if ( A failure exist in $C_{Config}$)
        ICF=true
        prrArray.add(RF)
    end if
    if ($historyAVE_{ND} \leq currentAVE_{ND}$)
        ICF=true
        prrArray.add(ND)
    end if
end if
if (ICF)
    $C_{Config}$ with
        sends a reConfig message for all headers in MANET
        Waiting for receive responses from other headers
        Calculate equation (17) for all headers and itself
        Create a list for all clusters based on equation (17) value
            and sort it by ascending order
if $C_{Config}$ is located in last half of the list
        ReConfig($C_{Config}$ , prrArray)
else
        ReOrgenize($C_{Config}$, prrArray)
end.
```

To this end, any service violation is a sign of a change in the current configuration of a cluster. But when is the actual reconfiguration?

When the current average of one required resource in C_{Config} is greater than or equal to its historical average or a accepted service is violated by it, C_{Config}, after collecting data from the other clusters make up a sorted list based on the Eq. (17). Now, based on where C_{Config} is in this list, two situations may occur:

a-C_{Config} is in the top half of the list.

b-C_{Config} is in the bottom half of the list.

By algorithm 5, After C_{Config} is located in the list, reorganization (Algorithm 6) or reconfiguration (Algorithm 7) occurs.

If $\mathbf{C_{Config}}$ *falls into the top half of the list*, the method maintains the cluster because of its relatively good performance compared to the other clusters. Suppose a service in C_{Config} is rejected or violated due to a lack of resource i, because the clusters know each other's position in MANET. In that case, C_{Config} may now allow its additional load to transfer and run its service in the nearest cluster with sufficient resource i. This license is, of course, valid until C_{Config} is in a position to execute the accepted service jointly. If, for any reason, C_{Config} violates these services types, this license will be revoked. It should be recalled that this outsourcing of service execution in some clusters is done to maintain high performance clusters.

If node failures and long-distance nodes are C_{Config} service violation reasons, due to the relatively good performance of clusters in the first half of the list, it is sufficient to evaluate the nodes of that clusters according to Eq. (3) and re-prioritize them.

```
Algorithm 6: ReOrgenize (C $C_{Config}$ , Array prr)

If ($M_i$ in prr)
  $C_{Config}$ with
    $C_{Cooperator}$ = finds a cluster in it's neighbor that has sufficent resource i
    Transfers additional load to $C_{Cooperator}$
If (RF or ND are in prr)
    $C_{Config}$ with
        Reprioritizes it's nodes base on equation (3)
        Rebuilds the cluster
```

If $\mathbf{C_{Config}}$ *is placed at the bottom half of the list*, the reconfiguration is done in this situation. After calculating Eqs. (18) to (22), C_{Config} broadcasts a reconfiguration message with its geographical location. Nodes in C_{Config} range (its members and non-members), upon receiving C_{Config} message, send a message containing their free amount of resources, trust value, and geographical location to C_{Config}. Then C_{Config} puts the obtained data from the nodes into a relation such as Fig. 5, which is called the SNR (Suitable Node Relation) Where M_{ij} is the amount of free resource j in node i, φ is trust degree of node i and DFH_i is the distance of node i from the C_{Config} header.

Suppose that by the occurrence of an event (rejection or violation of service) in the C_i cluster, that cluster is ready to be reconfigured after receiving the required data from the other clusters and with respect to being located at the bottom half of the list. In this state, C_{Config} broadcasts its reconfiguration message to attract new nodes. Free MANET nodes and other clusters' nodes that exist in C_{Config} range Upon receiving this message, if they wish to join this cluster, they send their data that containing free resources, trust, and geographic location in response to C_{Config}. After receiving the responses, C_{Config} inserts them into rows of SNR based on different fields. Now, based on the event that triggered the reconfiguration process, algorithm 7 run:

Fig. 5 SNR with 3 resources

	M_1	M_2	M_3	φ	DFH
1					
2					
3					
			⋮		
n					

Algorithm 7: ReConfig (C C_{Config} , Array prr)

```
 $C_{Config}$ with
     Calculates equations (18)-(22) and sorts their'values in ascending order
     Broadcasts a reConfig message for all nodes in own range
     SNR= Waits for responses in a given time
selectedNodes=null
 if ($\exists M_i \in prr$)
   selectedNodes=Select a node from
                   (Select * from SNR s
                      where ( $\forall M_i \in prr$, $C_{Config}.M_i < s.M_i$)
                        and
                           ( s.DFH < $C_{Config}$.DF)
                        and
                           (s.$\varphi$> $C_{Config}$.RF)
                                ) where (order of given equations values are ascending)
    if (selectedNodes = null)
       selectedNodes =Select top n row from
                        (Select * from SNR s
                           where  ( s.DFH < $C_{Config}$.DF)
                               and
                                 (s.$\varphi$> $C_{Config}$.RF)) order by
                                      $(0.5 \pm \alpha)\varphi + (0.5 \pm \alpha)DFH$   DESC
 else if ($\exists RF \in prr$)
    $C_{Config}$dismisses $n_f$    // $n_f$ is failured node
    selectedNodes=select a node from SNR s
                              where (max(s.$\varphi$))
                                   and
                                 ($\forall M_i \in prr$, $n_f.M_i \le s.M_i$)
                                   and
                                 ( s.DFH $\le n_f$.DFH)
    if (selectedNodes = null)
      selectedNodes =Select top n row from
                      (Select * from SNR s
                          where  ( s.DFH < $n_f$.DF)
                              and
                               (s.$\varphi$> $n_f$.RF)) order by s.$\varphi$  DESC
else if ($\exists ND \in prr$)
     $C_{Config}$ provides nodes' location to own subHeaders
     Subheaders calculate distance of each suitable node from itself and give that data to header
     $C_{Config}$ sort the nodes base on equation (25)
    while ($C_{Config}.M_i$-selectedNodes.$M_i$>0)
         selectedNodes= Select top n row from SNR s order by $\frac{s.\varphi}{s.DFH+DFAN}$
selectedNodes  join to $C_{Config}$
```

(a) The Reconfiguration Reason is Requests Reject due to Lack of Resource i: In this case, the needed current average of resource i to be higher than the needed average history of that resource in C_{Config}. So in SNR looking for a node that eliminates the need C_{Config} to resource i. Here's how to select nodes to join C_{Config} is explained: First, the method looks for nodes from SNR that the amount of each resource in it is greater than the value of the same resource in C_{Config}. Also, in those nodes, the DFH value must be lower than the average ND in C_{Config} and the φ value must be higher than the average RF in C_{Config}. Among the selected nodes, the method looks for a node to join C_{Config} that Observes prioritization between Eqs. (18) to (22). To more illustrate, the following example is presented:

Suppose, after calculating Eqs. (18) to (22) with the three sources, their priority is as follows:

$$M_3 < M_1 < RF < M_2 < ND \tag{23}$$

And the value of parameters in C_{Config} is as follows:

$$M_1 = 15 \cdot M_2 = 20 \cdot M_3 = 9 \cdot RF = 2 \cdot ND = 17$$

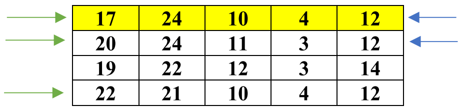
Given the values of these parameters, four nodes in the SNR are assumed that they are eligible to join C_{Config}. These four nodes are shown in Fig. 6. In these four nodes, DFH is smaller than ND, φ is larger than RF and the amount of all resources in those nodes is larger than the required amount of those resources in C_{Config}. Of these four nodes, given the higher priority of ND in the example, the method is looking for nodes that have at least DFH. Given this condition, rows 1, 2, and 4 have the smallest amount of DFH shown in Fig. 6 with the green arrow. Since M_2 is the next important parameter in the example, of these three rows, rows 1 and 2 have the highest value of this type and row 4 is deleted. Selected rows are shown in Fig. 6 with the blue arrow. Given the importance of the RF parameter in the next step, from rows 1 and 2, the choice of C_{Config} will be row 1. As a result, the row shown in yellow is selected as the appropriate node to join C_{Config}.

This way, not only the main cause of the reconfiguration, i.e., the need for resource i, will be eliminated, but with taking into account the sequence of possible reconfiguration factors, the reconfiguration will less happen.

If a node that alone cannot meet C_{Config}'s need for resource i does not exist, this is done: First, the rows of SNR whose DFH is less than the average ND in C_{Config} and their φ is larger than the average RF in C_{Config} are selected. Then selected rows according to the priority of the parameters in question are arranged and from the beginning of this sorted list, a number of nodes that satisfy C_{Config}'s need for resource i are selected. For example, based on the example (23) and because ND is more important than RF and RF is in the third priority, using the Eqs. (24), they sort in descending order. Then from the beginning of the list, enough nodes to satisfy C_{Config}'s need for resource i are selected.

$$(0.5 \pm \alpha)\varphi + (0.5 \pm \alpha)DFH \cdot (0.5 + \alpha) + (0.5 - \alpha) = 1 \tag{24}$$

In example (23) due to RF being in the third priority, α has a value of -0.3 and for DFH it has a value of + 0.3. It should be noted that since the number of prioritizing parameters in this study is 5, the α is $0 \le \alpha \le 0.5$ and, its sign can be positive or negative based on parameter priority.



17	24	10	4	12
20	24	11	3	12
19	22	12	3	14
22	21	10	4	12

Fig. 6 Four rows with Eligible to join CConfig and selecting the most appropriate node to join it

(b) The Reconfiguration Reason is a Violation of the Running Service Due to a Node Crash of the Cluster

The node that violates the running service certainly has provided some of the needed resources to run the service in the cluster. For this purpose, by removing that node from the cluster, the head uses node(s) of SNR that, in addition to the highest φ, fix the resource shortage of the removed node and their distance from the header is less than the distance of the removed node from the header.

(c) The Reconfiguration Reason May be a Violation of the Running Service Due to the Long Distance Between Cluster Nodes

If the reconfiguration is due to long nodes spacing, the C_{Config} header will provide this data to all of its subdomains after receiving the location of non-member nodes in its scope. The sub-headers then examine the distance of each of those nodes to the nodes below their cluster. If the distance of each non-member node from the header of a sub-cluster is smaller than the distance of the sub-cluster' nodes from the sub-cluster' header, sub-cluster' header sends a message to the C_{Config} header to capturing those non-member nodes. The C_{Config} header now arranges all the member and non-member nodes to which the capture message is sent, according to the Eq. (25) in descending order. It then absorbs enough nodes from the top of the sorted list to satisfy the required resources of that cluster in all types of resources.

$$\frac{\varphi_i}{DFH + DFAN} \tag{25}$$

DFAN is the distance of a non-member node from the sub-header of the cluster for which the capturing message was exported.

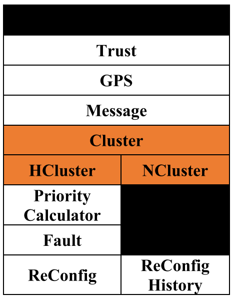
7 MANET Management Middleware

MMF to efficiently manage the clusters in MANET is created. MMF acts as the needed package for each node to participate in MANET. In other words, for a node to be recognized as a MANET member, it must have MMF in its main memory. If the node is joining MANET for the first time, it will request this middleware at the beginning of its entry into MANET. But if a node has a record of being in MANET, it calls MMF from its secondary memory to the main memory. Figure 7 shows the MMF. The various parts of MMF are mentioned in the previous sections.

8 Simulation and Performance Evaluation

In this section, the performance of the proposed clustering method for MANET, which we call MMF Clustering, with the WCA method [41] as a basic one-hop clustering method in MANET is compared. The simulation was performed by MATLAB R2016a on a Lenovo laptop with a core i5 processor, RAM 8, and 64-bit Windows 10 operating system. The intended area for the simulation is 500×500 square meters. Simulations were performed for one hour for each generated workload. In order to evaluate the proposed method, we determined three workloads with 250, 500 and 750 nodes. Then, from each workload, three samples were randomly generated. For each work in each population, the simulation

Fig. 7 MANET Management middleware



environment was performed for one hour and the average results of each workload were compared with another method. In other words, the simulation took 9 h. It is assumed that all the nodes in the environment are equipped with Wi-Fi and GPS to interact with each other and they randomly distributed in the simulation environment. Also, it is assumed that the nodes are free to move in any direction and remain in the environment until their battery is completely discharged or the node is damaged. In order to the simplicity of simulate, all nodes are present in the simulation environment at different speeds from the beginning of the simulation. The nodes can also stop at any time and stay without movement in the environment for a while. Depending on the speed of the various nodes in the environment, each node is allowed to move up to 100 m in one minute from its current location in any direction. It is also assumed that the communication range of each node is 30 m. In other words, each node directly covers a circle with a radius of 30 m. For each node, up to 5 resource types are provided that each node must have at least two resources of CPU and memory. Assume the memory available in each mobile node is between 8 and 128 GB and the processing capacity of each is between 250 and 1000 MIPS. The minimum and maximum wait times for each resource in a mobile node are set to 0 to 60 s based on the current load of each node. The discharge time of each mobile node is another parameter that is mentioned in this study. For simulation convenience, since the start of a node in the environment, which is the beginning of the simulation, the discharge start time of the node is considered. Also the node charging time below 10% as the end of node charging is considered. However, if the node increases its battery charge to 10% or more using a fixed and portable charging device and discharging starts again, the process of increasing the charge and restarting the battery discharge will consider as the node's discharging next step (10% is intended for the owner of the mobile node to re-charge the device before shutting the node off and exiting from MANET). By examining the actual mobile nodes in the surrounding environment and considering that the battery discharge rate at different times is related to various hardware and software factors, hence to facilitate simulation, depending on load of a mobile node, the discharge rate of each node between 1 to 3 percent per minute is considered.

Each node is able to send requests with the desired specifications to MANET when it is in the environment. Each service request can include the maximum resources available per node and even more. The maximum waiting time and the maximum run time are designated 60 and 300 s for each service respectively. The following criteria to evaluate the performance and compare methods have been used.

8.1 The Amount of Fulfilled QoS

As shown in Figs. 8, the proposed method performs much better in QoS than WCA. These diagrams show that the more nodes in the environment, this performance will be maintained again. This result is due to the fact that in the proposed method, unlike the previous method, when selecting or forming clusters, it also considers the requirements of the cluster in addition to meeting the current service requirements. The proposed method seeks to maintain and manage the current cluster needs in a way that can accommodate more requests and provide QoS at a higher level. As a result, in the proposed method, we have achieved this by timely detecting clusters performance decline, reorganizing and reconfiguring relatively inefficient clusters and removing clusters that are no longer hopeful of profitability in MANET.

Figure 8d also shows that re-configured clusters violate fewer requests after reconfiguration due to considering the cluster's current and future needs. In contrast, cluster configuration in the WCA due to non-consideration these indicators have been less successful in meeting subsequent requests.

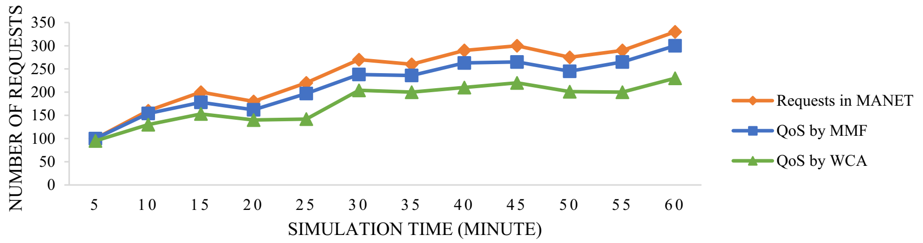
8.2 Load Balance in all Nodes in a Cluster, Amount of Wasted Resources in Different Nodes and Amount of Resource Usage in Request Node

Since mobile nodes in MANET have different resources at low scales, optimally utilizing these resources and preventing them from being wasted is one of the critical issues in this environment. In this regard, Figs. 9, 10 has been shown the proposed method results and WCA results, based on the amount of resource loss and load distribution. As is mentioned in the previous section, In the proposed method, in important decisions such as joining a node to a cluster or forming a cluster, nodes that need more of each other's resources are more likely to be together. In other words, the greater mutual need of the partner nodes for each other's resources will be fewer wasted resources.

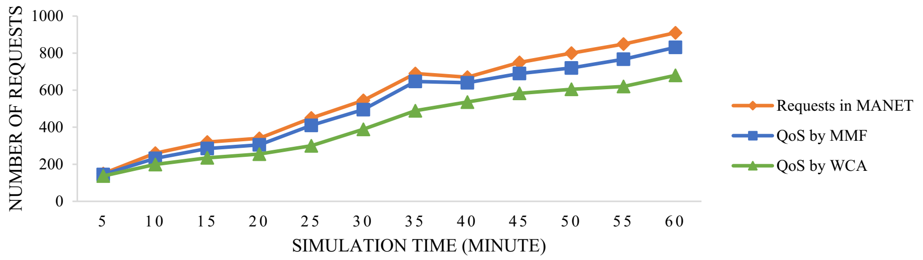
To calculate the amount of resources used by the applicant node, in the MMF, during the whole simulation time, the amount of resources used by each applicant node in its accepted cluster was studied. Figure 9a shows the average resource utilization of all service request nodes during the simulation period in two compared methods. In MMF, due to the mutual need of applicant nodes and clusters for resources, the use of resources has been much better compared to WCA.

It was said that in the MMF, in each cluster, the stronger nodes in terms of resources and the level of trust take over the services, and the weaker nodes take over the management of messages. Hence, most nodes in each cluster are involved in performing tasks to advance the goals of that cluster. The result of this collaboration will be less waste of resources in MANET and more efficiency. Figure 9b shows the average unused resources per cluster in the MMF is significantly lower than the WCA.

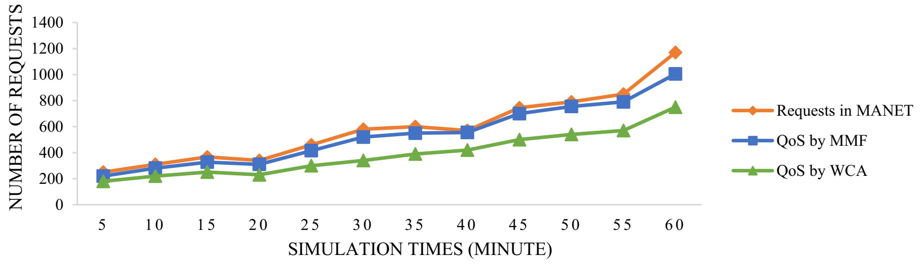
But in terms of uniform load distribution, MMF look for nodes that are less loaded. In other words, nodes with a higher current load will be less important to choose as a service host. Therefore, to express the uniformity of the load distributed in the clusters obtained from MMF compared to WCA, we performed the distribution of requests on 5 and 10 clusters. Figures 10a-d show the average distribution of requests during the simulation in each cluster for three-run for each workload.



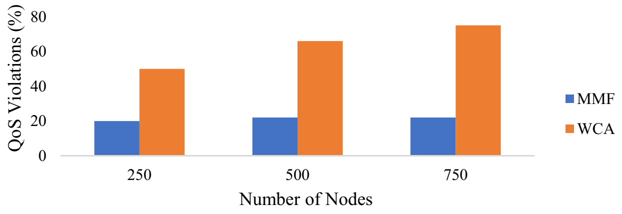
(a) Provided QoS with 250 nodes



(b) Provided QoS with 500 nodes



(c) Provided QoS with 750 nodes

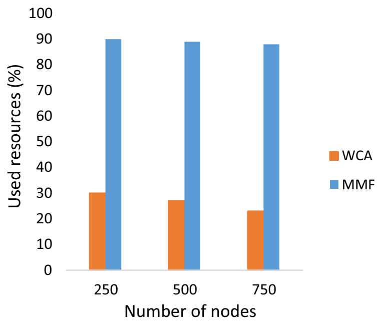


(d) Violation by Reconfigure/Reorganize Clusters

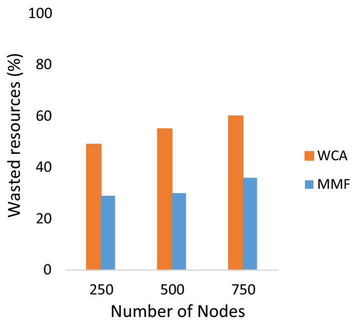
Fig. 8 The provided QoS in Comparative Methods

8.3 The Stability of the Clusters in MANET

In the present study, some strategies propose to deal with the reduction of cluster efficiency in MANET. These solutions are reorganization and reconfiguration. The reorganization is done on clusters with acceptable performance, but if there is no new order in place, they will face a severe decline in their performance. The reorganization purpose is to maintain



(a)



(b)

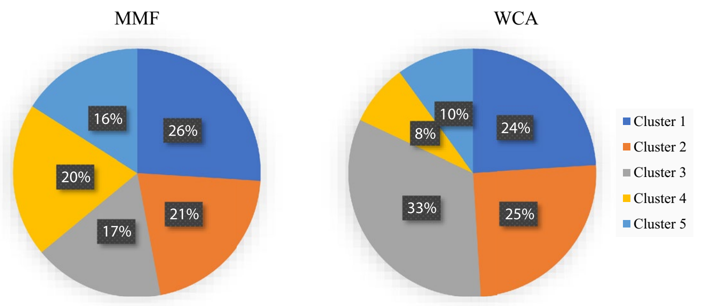
Fig. 9 **a** The average used resources from applicant node in the accepted cluster. **b** Average amount of wasted resources per cluster

the existing clusters by creating a new order in their nodes. Another solution, reconfigure, is dealing with clusters whose performance is greatly reduced and the solution is to enhance those clusters by removing some nodes and absorbing new nodes. It has been noted, however, that clusters that do not hope to increase their efficiency will be permanently removed from MANET by MMF. By MMF, in reconfiguring a cluster, in addition to resolving the current clustering problem (reduced performance), potential issues that may occur in the not too distant future are considered too. Therefore, it is expected the clusters to be more stable in this way and to require fewer configurations.

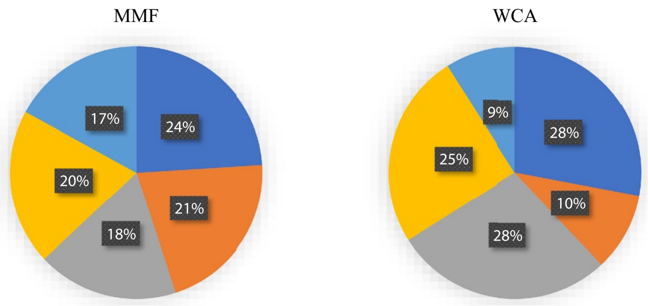
Of course, in WCA, like MMF, the reconfiguration is not done. In WCA, there is a concept called the dominating set that reconfiguration means a change in that set. For this reason, we have slightly changed the reconfiguration in the WCA to be closer to the reconfiguration proposed in MMF. In order to evaluate the stability of clusters in the two compared methods, simulations were performed several times for 250 nodes with 25 10-node clusters, 500 nodes with 25 20-node clusters, and 750 nodes with 30 25-node clusters. The average stability of the clusters per load is illustrated in Fig. 11.

9 Conclusions and Future Works

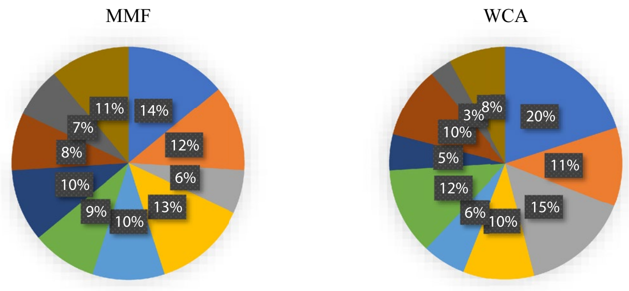
In this study, the issue of cluster management in MANET was addressed. We attempted to examine this topic in greater detail so that the reader would be familiar with most aspects of MANET cluster management. In this research, however, one-hop cluster management was checked out. To increase the efficiency of MANET, some clusters were maintained by permanently monitoring the parameters affecting the performance of the environment and were ended the work of others. Therefore, due to the constant change in the current state of the nodes, using greedy methods far from complexity, the clusters whose performance was declining during their lifetime in the environment were identified. Two methods of reorganization and reconfiguration to restore their performance were used. In order to effectively apply these methods, first, the MANET clusters were divided according to their efficiency into two groups. The first group, clusters whose performance was acceptable, but this performance had a decreasing slope. By identifying the causes of these clusters' performance loss and by reorganizing them, they were prevented from their performance loss.



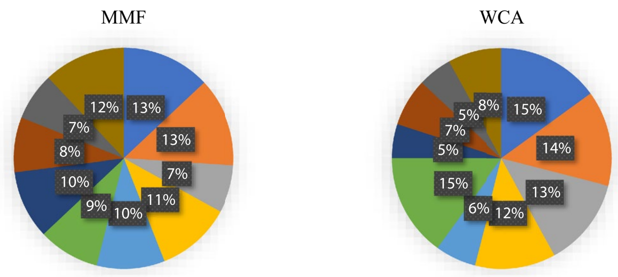
(a) Average load distribution in 5 clusters with 250 Nodes and 150 requests



(b) Average load distribution in 5 clusters with 500 Nodes and 250 requests

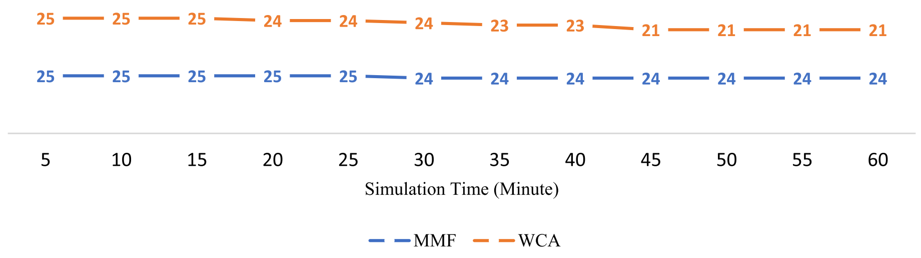


(c) Average load distribution in 10 clusters with 500 Nodes and 450 requests

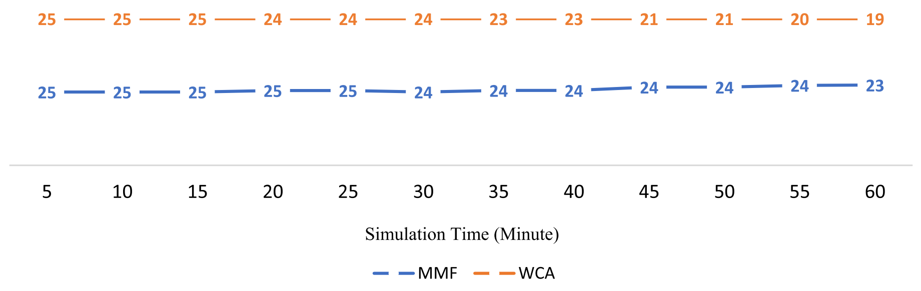


(d) Average load distribution in 10 clusters with 750 Nodes and 500 requests

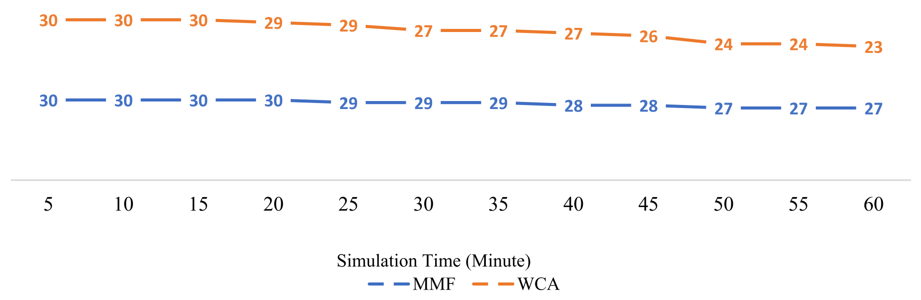
Fig. 10 Load balance in different workloads



(a) 250 nodes with 25 10-node clusters



(b) 500 nodes with 25 20-node clusters



(c) 750 nodes with 30 25-node clusters

Fig. 11 Average stability time of clusters during simulation (60 min) at different number of nodes

But there were other clusters that greatly were reducing MANET's performance. These types of clusters could no longer progress on the basis of their current nodes in line with the MANET target. To this end, by reconfiguring and using a new combination of nodes in the environment formed clusters with good performance. However, clusters that could not even be reconfigured to increase the efficiency of the environment were identified and terminated.

For future work, we plan to expand our work with multi-hops cluster management and look for a faster way to manage dynamic data in the environment. So whenever data was needed to make decisions, can access to them in the shortest time possible. We also provide a way to deal with possible clustering failures, so that in the event of any disruption to the execution of the services, MANET can efficiently manage the failures using an agile method to the extent possible for the services.

Acknowledgements The authors are grateful to all study participants.

Author Contributions Iraji and sokhtsaraei suggested the algorithm for image analysis; sokhtsaraei implemented it and analyzed the experimental results; Tanha provided clinical guidance; Iraji, sokhtsaraei, and Nejadkheirallah consulted the obtained result. All authors read and approved the final manuscript.

Funding This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflicts of interest The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Ethical Approval This article does not contain any data, or other information from studies or experimentation, with the involvement of human or animal subjects.

Consent for Publication Agreed by the authors.

References

1. Cooper, C. (2012). Apples cook: 172 million post-PC devices in the last year,'' CNET, New York, NY, USA, Tech. Rep., Mar.
2. Ashwin, M., Kamalraj, S., & Azath, M. (2016). Weighted clustering trust model for mobile Ad Hoc networks. *Wireless Personal Communication, 94*(4), 1–10.
3. Chen, I. R., & Guo, J. (2015). Hierarchical trust management of community of interest groups in mobile ad hoc networks. *Ad Hoc Networks, 33*, 154–167.
4. Xu, L., Wang, J., Liu, Y., Shi, W., & Gulliver, T. A. (2018). Outage performance for IDF relaying mobile cooperative networks. *Mobile Networks & Applications, 23*(6), 1496–1501.
5. Tarique, M., Tepe, K. E., Adibi, S., & Erfani, Sh. (2009). Survey of multipath routing protocols for mobile ad hoc networks. *Journal of Network and Computer Applications, 32*(6), 1125–1143.
6. Waluyo, A. B., Taniar, D., Rahayu, W., & Srinivasan, B. (2017). Trustworthy data delivery in mobile P2P network. *Journal of Computer and System Sciences, 86*, 33–48.
7. Bhardwaj, A., Al-Turjman, F., Kumar, M., Stephan, T., & Mostarda, L. (2020). Capturing-the-invisible (CTI): behavior-based attacks recognition in iot-oriented industrial control systems. *IEEE Access, 8*, 104956–104966.
8. Singh, A. K., Alshehri, M., Bhushan, S., Kumar, M., Alfarraj, O., & Pardarshani, K. R. (2021). Secure and energy efficient data transmission model for WSN. *Intelligent Automation & Soft Computing, 27*(3), 761–769.
9. Waluyo, A. B., Taniar, D., Rahayu, W., Aikebaier, A., Takizawa, M., & Srinivasan, B. (2013). Mobile peer-to-peer data dissemination in wireless ad-hoc networks. *Information Sciences, 230*, 3–20.
10. IEEE 802.15 Working Group for WPAN. (2011). from http://www.ieee802.org/15/.
11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (2007). IEEE Standard 802.11, IEEE Computer Society.
12. Ryu, JH., Song, S., Cho, DH. (2001). New clustering schemes for energy conservation in two-tiered mobile ad-hoc networks. ICC 2001 IEEE International Conference on Communications, 862_866.
13. Lacuesta, R., Penalver, L., Fernandez-Sanz, L., Lloret, j., Garcia, M. (2009). Software Requirements for Ubiquitous Ad Hoc Mobile Networks: An Example of a Bluetooth Application. International Conference on Software Engineering Advances. IEEE, 179–184.
14. Arunachalam, A., & Sornil, O. (2015). Issues of implementing random walk and gossip based resource discovery protocols in P2P MANETs & suggestions for improvement. *Procedia Computer Science, 57*, 509–518.

15. Sarma, A. D., Molla, A. R., & Pandurangan, G. (2015). Efficient random walk sampling in distributed networks. *Journal of Parallel and Distributed Computing, 77*, 84–94.
16. Pu, I. M., Stamate, D., & Shen, Y. (2014). Improving time-efficiency in blocking expanding ring search for mobile ad hoc networks. *Journal of Discrete Algorithms, 24*, 59–67.
17. Xu, D., Nahrstedt, K., Wichadakul, D. (2001). QoS-Aware Discovery of Wide-Area Distributed Services. Proceedings First IEEE/ACM International Symposium on CLUSTER Computing and the Grid, 92–99.
18. Liang, J. C., Chen, J. C., & Zhang, T. (2011). An adaptive low-overhead resource discovery protocol for mobile ad-hoc networks. *Wireless Networks, 17*(2), 437–452.
19. Hongyan, M., Yujie, Z., Xiangwu, M. (2014). A path tracking search algorithm based on the credibility of node service ability. IEEE Conference on Wireless Communications and Networking, 3385– 3389.
20. Liu, H., & Zhang, X. (2017). Efficient resource search mechanism in selfish mobile peer-to-peer network. *Journal of system simulation, 29*(5), 1093–1102.
21. Mondal, A., Madria, SK., Kitsuregawa, M. (2006). CLEAR: An Efficient Context and Location-Based Dynamic Replication Scheme for Mobile-P2P Networks. 17th International Conference on Database and Expert Systems Applications, Springer-Verlag, 399–408.
22. Kantere, V., Tsoumakos, D., Sellis, T., & Roussopoulos, N. (2009). GrouPeer: dynamic clustering of P2P databases. *Information Systems, 34*(1), 62–86.
23. Seddiki, M., & Benchaïba, M. (2016). 2P-lookup: popularity and proximity based P2P lookup mechanism over MANETs. *Journal of Network and Computer Applications, 71*, 181–193.
24. Gupta, P., & Kumar, P. R. (2000). The capacity of wireless networks. *IEEE Transactions on Information Theory, 46*(2), 388–404.
25. Hong, X., Xu, K., & Gerla, M. (2002). Scalable routing protocols for mobile ad hoc networks. *IEEE Network, 16*(4), 11–21.
26. Xu, K., Hong, X., Gerla, M., (2002). An ad hoc network with mobile backbones. IEEE International Conference on Communications (ICC), 3138_3143.
27. Belding-Royer, E. M. (2002). Hierarchical routing in ad hoc mobile networks. *Wireless Communications and Mobile Computing, 2*(5), 515–532.
28. Perkins, CE., (2001). Ad Hoc Networking. Reading, MA, USA, Addison-Wesley.
29. Basagni, S., Chlamtac, I., (1997). A Generalized Clustering Algorithm for Peer-to-Peer Networks. in Workshop Algorithmic Aspects of Communication, 1–15.
30. Ergenç, D., Eksert, L., & Onur, E. (2019). Dependability-based clustering in mobile Ad-Hoc networks. *Ad Hoc Networks, 93*, 101926.
31. Yang, Z., Wu, W., Chen, Y., Lin, X., Chen, X., (2018). Navigation Route Based Stable Clustering for Vehicular Ad Hoc Networks. International Conference on Communications and Networking in China, 552–562.
32. Han, T., Zhang, L., Pirbhulal, S., Wu, W., & de Albuquerque, V. H. C. (2019). A novel cluster head selection technique for edge-computing based IoMT systems. *Computer Networks, 158*, 114–122.
33. Chaudhry, Ch., & Tapaswi, Sh. (2018). Optimized power control and efficient energy conservation for topology management of MANET with an adaptive Gabriel graph. *Computers & Electrical Engineering, 72*, 1021–1036.
34. Nabar, K., & Kadambi, G. (2018). Affinity propagation-driven Distributed clustering approach to tackle greedy heuristics in mobile Ad-hoc networks. *Computers and Electrical Engineering, 71*, 988–1011.
35. Bhushan, Sh., Kumar, M., Kumar, P., Stephan, Th., Shankar, A., & Liu, P. (2021). FAJIT: a fuzzy-based data aggregation technique for energy efficiency in wireless sensor network. *Complex & Intelligent Systems, 7*, 997–1007.
36. Rao, M., & Singh, N. (2018). Energy efficient QoS aware hierarchical KF-MAC routing protocol in manet. *Wireless Personal Communications, 101*(2), 635–645.
37. Ahmad, M., Hameed, A., Ullah, F., Wahid, I., Rehman, S. U., & Khattak, H. A. (2018). A bio-inspired clustring in mobile adhoc networks for internet of things based on honey bee and genetic algorithm. *Journal of Ambient Intelligence and Humanized Computing, 11*, 4347–4361.
38. Meng, X., & Deng, Y. (2019). A time-aware resource search strategy with the ant colony optimization in MANETs. *Peer-to-Peer Networking and Applications, 12*(5), 1013–1027.
39. Chithaluru, P., Al-Turjman, F., Kumar, M., & Stephan, Th. (2020). I-AREOR: An energy-balanced clustering protocol for implementing green IoT in smart cities. *Sustainable Cities and Society, 61*(8), 102254.
40. Srivastava, Sh., Saxena, S., Buyya, R., Kumar, M., Shankar, A., & Bhushan, B. (2021). CGP: Cluster-based gossip protocol for dynamic resource environment in cloud. *Simulation Modelling Practice and Theory, 108*, 102275.

41. Das, S., Chatterjee, M., & Turgut, D. (2002). WCA: a weighted clustering algorithm for mobile Ad Hoc networks. *Cluster Computing.*, *5*(2), 193–204.

Reza Sookhtsaraei is faculty member of Department of Computer Engineering and bio Technology, Payame Noor University, Iran. He has master degree in computer science. He published some 10 papers in professional journals.

Mohsen Nejadkheirallah received the Master degree in computer science from Payame Noor University of Tehran and is currently a PhD candidate in Islamic Azad University, Babol Branch. His research interests includedata analysis, data mining algorithms and applications, machine learning and wireless sensor networks.

Mohammad Saber Iraji is faculty member of Department of Computer Engineering and bio Technology, Payame Noor University, Iran. He has master degree in computer science. He published some 30 papers in professional journals.