Energy-Efficient Clustering-Based Byzantine Fault Tolerance Algorithm in Manet (EECBFT)

A. MAHENDRAN[1], Dr. C. KAVITHA[2], Dr. K. SAKTHIVEL[3]
[1] *Ph.D Research Scholar, Dept of Computer Science, Periyar University, Salem*
[2] *Assistant Professor, Dept of Computer Science, Thiruvalluvar Govt Arts College, Rasipuram*
[3] *Professor, Dept of CSE, KSR College of Technology, Tiruchengode*

Abstract - Mobile Ad-Hoc Networks (MANETs) have witnessed significant growth, driven by the proliferation of wireless-enabled portable devices. This article explores critical aspects of MANETs, including cluster formation, fault tolerance, and the innovative concept of Energy-Based Byzantine Fault Tolerance (BFT). Cluster formation is essential for efficient communication and resource management in MANETs. This article delves into proximity-based clustering, node distance calculation, and cluster head election to create robust communication structures. Fault tolerance is paramount in MANETs, where node failures can occur due to various factors. Clustering-based fault tolerance is discussed, emphasizing fault detection, isolation, and redundancy to ensure network resilience. The article introduces the concept of Energy-Based Byzantine Fault Tolerance (BFT) as a novel approach to address the unique challenges of MANETs. Energy-aware strategies are integrated with Byzantine Fault Tolerance to enhance network reliability and energy efficiency. The Energy-Based BFT algorithm aims to optimize cluster formation, Byzantine fault tolerance, energy-efficient communication, dynamic energy management, energy-aware quorum selection, fault detection with energy conservation, and head election based on energy reserves. The expected outcomes of this research include developing and validating the Energy-Based BFT algorithm, tailored for MANET clustering, to improve fault tolerance, energy efficiency, and network reliability. The article highlights the promising potential of Energy-Based BFT in fortifying MANETs against malicious attacks while extending their operational lifespan. Further research and empirical evaluations are suggested to fully understand its practical implications.

Indexed Terms- MANET, Fault tolerance, BFT, Energy Efficiency

I. INTRODUCTION

The growth of wireless-enabled devices has driven Mobile Ad-Hoc Networks (MANETs) characterized by mobile nodes freely establishing wireless connections. MANETs adapt to changing network conditions and are essential in infrastructure-limited areas. Clustering, a core MANET feature, promotes scalability and efficiency.

In MANETs, efficient cluster formation is crucial for reducing routing overhead and enhancing network management. Clusters, led by cluster heads, facilitate communication within their boundaries. Cluster heads also manage intra-cluster communication, while gateway nodes enable inter-cluster communication.

Clustering offers several advantages:

1. Improved system capacity through resource reuse within clusters.
2. Reduced routing information and delays via a virtual backbone created by cluster heads.
3. Minimized data for representing network state.
4. Reduced topology update messages by updating information within clusters.

II. FAULT TOLERANCE

Fault tolerance is vital in MANETs due to node failures caused by factors like battery depletion or hardware issues. Clustering-based fault tolerance leverages cluster structures to enhance network resilience. Key steps include cluster formation, fault detection, fault isolation, and redundancy with backup nodes.

Clustering-Based Fault Tolerance

Clustering-based fault tolerance is an approach that leverages the hybrid insights of cluster structure in a MANET to enhance the network's resilience to node failures.

Cluster Formation: Initially, the network is organized into clusters. Each cluster has a head or cluster head responsible for coordinating communication within the cluster.

Fault Detection: Nodes within a cluster monitor the status of their neighbors. If a node detects a neighbor's failure or fault (e.g., no response to communication), it reports the fault to the cluster head.

Fault Isolation: Upon receiving fault reports, the cluster head can isolate the faulty node or adjust routing paths to avoid it. This prevents the faulty node from affecting the entire network.

Redundancy: clustering-based fault tolerance may involve redundant cluster heads or backup members. If the primary cluster head fails, these backup nodes can take over the responsibilities of the cluster head.

III. EXISTING METHODOLOGY

A. Improved Performance Clustering Using Modified K-Means Algorithm

Amit Gupta, Dr. Mahesh Motwani (2021) et.al proposed a network performance enhancement; clustering has been identified as a valuable technique since it limits the active participation in routing processes to cluster heads exclusively. This approach involves the random distribution of network nodes, which are subsequently grouped into clusters by applying a modified K-means clustering methodology. This research introduces an innovative method for initializing centroids within the K-means algorithm. Here, we utilize Geographic Centers as the initial centroids for cluster formation. Furthermore, the paper addresses the crucial task of cluster head selection, which plays a pivotal role in network optimization. To accomplish this, we employ the weighted multi-criterion acceptability method, which considers various performance metrics for cluster head designation. This selection method leads to substantial improvements in network performance, specifically concerning Load Balancing Factor, PDF, and Throughput.

B. Trust Value Updation Algorithm

Sapna B. Kulkarni (2017), et.al proposed a Trust Value Update Algorithm for Multicast Routing in Cluster-Based MANETs. The selection of a cluster head plays a crucial role in the network. However, suppose the chosen cluster head node becomes selfish or malicious. In that case, it can negatively impact the overall cluster communication, leading to a decrease in the throughput and efficiency of the MANET. Therefore, trust needs to be established for all nodes within the cluster, making trust management essential. In contrast to previous methods, the proposed algorithm reduces energy consumption and minimizes delays. It ensures the cluster head is trustworthy and avoids selecting malicious, selfish, or dark-hole attacker nodes as cluster heads. Hence, trust management is vital in this context. The trust value of nodes within the cluster is determined using a challenge assessment algorithm to identify malicious nodes. Trust or trust value represents the extent to which one can rely on or have confidence in a node. Nodes whose trust value falls below the threshold are added to the blacklist. The cluster head selection considers the node's trust value and parameters such as bandwidth, power consumption, and network connectivity. The trust value of nodes should be continuously monitored over a specified period. If the trust value falls below the threshold, the node is identified as malicious and removed from the blacklist.

C. Fuzzy Approach for Secure Clustering in MANET

Kosuke Ozera (2017), et.al proposed an approach using fuzzy logic to enhance secure clustering in Mobile Ad hoc Networks (MANETs) and studied the effects of the distance parameter on system performance. MANET is a wireless network where mobile nodes are dynamic, have limited bandwidth, and operate on minimal battery power. Clustering is the process of grouping mobile nodes, and it involves selecting a head node to manage the entire network. Various clustering schemes have been introduced, including Mobility-based, Energy-efficient, Connectivity-based, and Weighted-based. The authors then presented and compared two fuzzy-based frameworks, namely F2SMC1 and F2SMC2, to improve the security of cluster nodes in MANETs. They examined the performance of F2SMC1 and F2SMC2 and demonstrated that F2SMC2 exhibits greater effectiveness in terms of its degree of fuzziness compared to F2SMC1. Additionally, F2SMC2 showed better management of nodes within the cluster when compared to F2SMC1. This algorithm effectively reduces power consumption, resulting in an extended network lifetime.

D. Improving Reliability of Cluster Nodes in MANETs: A Fuzzy-Based Approach

Mirjeta Alinci (2016), et.al proposed a Fuzzy-Based Approach to improve the reliability of cluster nodes in Mobile Ad hoc Networks (MANETs). MANET is a wireless network where mobile nodes are dynamic, have limited bandwidth, and operate on minimal battery power. Mobile nodes can be grouped into clusters to enhance stability and scalability in challenging environments. Clustering involves selecting a head node to manage the entire network, and various clustering approaches focusing on different performance metrics have been introduced. Several clustering schemes have been presented: Mobility-based, Energy-efficient, Connectivity-based, and weighted clustering. The system can achieve higher reliability and improved performance by selecting nodes with low Packet Loss (PL) rates. In this type of network, each device acts as both a host and a router and can reconfigure itself. It can be rapidly deployed and reconfigured when a communication infrastructure is unavailable. Consequently, MANET faces several challenges, including host mobility, dynamic topology, multi-hop transmission, limited bandwidth, and battery constraints. Therefore, studying MANET is a challenging task. The network is divided into logical entities called clusters to enable efficient routing and monitoring. Within a cluster, the nodes responsible for coordinating cluster activities are called Cluster Heads (CH). CHs provide services to other nodes, while the remaining nodes are common or cluster members.

IV. PROPOSED ENERGY-EFFICIENT CLUSTERING BASED BYZANTINE FAULT TOLERANT (BFT) ALGORITHM FOR MANET

Mobile Ad Hoc Networks (MANETs) represent a dynamic and self-organizing paradigm for wireless communication, enabling the formation of networks without the reliance on a fixed infrastructure. These networks have found applications in many scenarios, from military operations and disaster management to Internet of Things (IoT) deployments. However, their unique characteristics, inherent mobility, and constrained energy resources pose significant challenges in ensuring reliable and secure communication.

The unreliability of wireless communication channels and the potential presence of Byzantine-faulty nodes, which may exhibit arbitrary and malicious behavior, are among the key challenges that MANETs face. Byzantine failures can disrupt network operations, compromise data integrity, and hinder consensus among network nodes. Therefore, the development of Byzantine Fault Tolerant (BFT) algorithms is critical for safeguarding MANETs against the perils of malicious actors and unreliable communication.

A. Problem statements

Mobile Ad Hoc Networks (MANETs) are characterized by their dynamic and self-organizing nature, making them susceptible to various challenges, including energy constraints and Byzantine failures. The problem involves the development of an Energy-based Byzantine Fault Tolerant algorithm tailored for MANET clustering. The algorithm aims to enhance the communication and consensus process's reliability and energy efficiency within MANET clusters.

B. Problem Description

In MANETs, nodes communicate without a fixed infrastructure, forming clusters to optimize network performance and resource management. However, the dynamic nature of MANETs and energy constraints present challenges in achieving fault tolerance and energy efficiency within clusters. The problem involves designing a BFT algorithm that incorporates energy-aware strategies and provides the following:

1. Energy-Aware Cluster Formation: Develop mechanisms for forming clusters within the MANET that consider nodes' energy levels and characteristics. Nodes with similar energy profiles may be grouped to optimize energy usage.
2. Byzantine Fault Tolerance: Design a robust Byzantine fault tolerance mechanism that tolerates Byzantine failures, including malicious node behavior, within MANET clusters. The algorithm should ensure reliable consensus even when Byzantine-faulty nodes are present.
3. Energy-Efficient Communication: Implement energy-efficient communication protocols that minimize energy consumption during message transmission and reception. Optimize routing and data aggregation to reduce energy overhead.

4. Dynamic Energy Management: Create adaptive energy management strategies that allow nodes to adjust their roles and responsibilities based on their energy levels. Ensure that energy-depleted nodes are not overburdened with tasks.
5. Energy-Aware Quorum Selection: Develop quorum-based decision-making processes that consider energy levels when selecting quorums. Minimize the energy expenditure required for achieving consensus within clusters.
6. Fault Detection with Energy Conservation: Implement fault detection mechanisms that efficiently identify Byzantine-faulty nodes without imposing excessive energy overhead on the network.
7. Head Election and Energy Reserves: Design head election processes considering nodes' energy reserves. Ensure that cluster heads are selected based on energy availability to maintain cluster operations.

C. Proposed Methodology

- To develop an Energy-Efficient Clustering-based BFT algorithm customized for MANET clustering.
- To enhance the reliability of MANET clusters while considering energy constraints.
- To optimize energy usage in communication and consensus processes within clusters.
- To extend network lifetime and sustainability by conserving energy resources.

D. Expected Outcomes

The expected outcomes of this research are the development and validation of an Energy-Efficient clustering-based BFT algorithm specifically tailored for MANET clustering. The algorithm should demonstrate improved fault tolerance, energy efficiency, and reliability within MANET clusters.

E. Manet cluster formation

Mobile Ad Hoc Networks (MANETs) are dynamic wireless networks consisting of mobile nodes that communicate with each other without a fixed infrastructure. These networks are highly adaptable and find applications in various scenarios, including military operations, emergency response, IoT deployments, etc. Cluster formation plays a pivotal role in managing the inherent challenges of MANETs, such as limited resources, network scalability, and dynamic topologies. MANETs are characterized by their decentralized nature, where each node can communicate directly with any other node within its communication range. However, this unrestricted communication can lead to high overhead, inefficient resource utilization, and increased energy consumption. Cluster formation addresses these challenges by organizing nodes into groups or clusters, each with a designated leader or cluster head. This hierarchical structure streamlines communication, reduces overhead, and enhances network efficiency.

F. Neighbor Discovery Protocol

Neighbor Discovery Protocol (NDP) is a fundamental component of Mobile Ad Hoc Networks (MANETs), including within the context of clustering. NDP is responsible for nodes in the network discovering and maintaining information about their neighboring nodes. Here's how NDP can be used in the context of clustering in a MANET:

1. Neighbor Detection:
NDP helps nodes detect and identify their neighboring nodes within their communication range. This is crucial for clustering as nodes need to know which other nodes are nearby to form clusters effectively.

2. Cluster Formation:
Once nodes have identified their neighbors, they can use this information to initiate cluster formation. NDP assists in this process by providing information about potential cluster members.

3. Cluster Head Selection:
NDP can be utilized to facilitate the selection of cluster heads. Nodes can exchange information about their resources, such as available battery power, processing capability, and communication quality, through NDP messages. This information lets nodes decide which nodes should serve as cluster heads.

4. Dynamic Cluster Maintenance:
NDP continuously updates the information about neighboring nodes. In the context of clustering, this information can be used to dynamically adapt the cluster structure. For example, if a node with a higher battery power level becomes a neighbor, it might be considered as a potential cluster head.

5. Cluster Communication:
NDP ensures that cluster members maintain connectivity with each other. It helps nodes within the same cluster exchange control messages and data efficiently.

6. Fault Tolerance:
NDP's neighbor information can also be used for fault tolerance. If a cluster head fails or loses connectivity with its cluster members, NDP can aid in identifying a suitable replacement or rerouting communication paths.

7. Energy Efficiency:
NDP messages can include information about nodes' energy levels. This information can be used in clustering algorithms to favor nodes with higher energy reserves as cluster heads to prolong the network lifetime.

8. Adaptation to Mobility:
MANETs are dynamic, and nodes may change their positions frequently. NDP continuously updates neighbor information, allowing clusters to adapt to changes in network topology due to node mobility.

Neighbors Table
In a Mobile Ad Hoc Network (MANET) cluster, the "Neighbors Table" (NTAB) is a data structure or a table used by nodes to maintain information about their neighboring nodes within the cluster. The NTAB is an essential cluster management component and facilitates efficient communication and coordination among cluster members. Here is what you might typically find in a Neighbors Table for a MANET cluster:

1. Node Information:
Each entry in the NTAB includes information about neighboring nodes. This information may include:
Node ID or address: A unique identifier for each neighboring node.
Location: The physical or logical location of the neighboring node within the cluster.
Battery Power: The remaining energy level of the neighboring node's battery.
Communication Range: The maximum distance over which the neighboring node can communicate.
Mobility Status: Information about the mobility pattern of the neighboring node (e.g., static or mobile).

2. Cluster Role Information:
The NTAB may include information about the role of each neighboring node within the cluster. For example:
Cluster Head: Indicates whether a neighboring node is a cluster head.
Cluster Member: Indicates whether a neighboring node is a regular cluster member.
Relay Node: Indicates whether a neighboring node is a relay node within the cluster.

3. Communication Quality Metrics:
The NTAB may store metrics related to the quality of communication links with neighboring nodes. This could include signal strength, packet loss rates, and latency.

4. Cluster Topology Information:
Information about the connectivity and topology within the cluster. This may include details about how neighboring nodes are interconnected within the cluster.

5. Neighbor Status and Health:
Information about the status and health of neighboring nodes, including any recent connectivity or health issues. For example, this information might be recorded if a neighboring node has experienced communication problems or has a low battery.

6. Dynamic Updates:
The NTAB is typically dynamic and continuously updated as nodes move, join, or leave the cluster. Nodes exchange information periodically to keep the NTAB up to date.

7. Security Information:
Depending on the MANET's security requirements, the NTAB may also include information related to node authentication and trust levels for secure communication within the cluster. The Neighbor’s Table is critical in cluster management, helping nodes make informed decisions about routing, cluster head selection, and resource allocation. It provides the necessary awareness of neighboring nodes' characteristics and status, contributing to efficient and reliable communication within the MANET cluster.

Calculation of the Node weight

Node weight calculation is a critical aspect of the process. The movement patterns of individual nodes exert a notable impact on the overall system's stability, consequently influencing the network's topology. Selecting a limited number of highly stable, less mobile nodes is advisable to foster a robust virtual backbone. This choice bolsters the backbone's security and mitigates the risk of depleting energy resources in lightweight nodes when routing packets through them, which could result in link failures. In such instances, the routing path becomes disrupted, necessitating the establishment of alternative routes to handle failures, a topic explored in the relevant section.

In designing and managing Mobile Ad hoc Networks (MANETs), selecting Cluster Heads (CHs) and cluster members is pivotal in optimizing network performance and ensuring robustness. A key consideration in this process is the calculation of node weights, which help determine the suitability of individual nodes for specific roles. Node weight is typically derived from three critical factors: Energy Level (E), Mobility (M), and Connectivity (C). By assigning weight coefficients (alpha, beta, and gamma) to these factors, nodes can be ranked based on their suitability for CH roles or cluster membership. The Energy Level reflects a node's remaining energy reserves, crucial for energy-efficient routing and operation. Mobility captures a node's ability to move or adapt to network dynamics, influencing its role in maintaining cluster stability. Connectivity assesses a node's communication links within the network, affecting its ability to relay data. A weighted sum of these normalized factors determines each node's weight, enabling the selection of CHs or cluster members that best align with network goals, whether it is optimizing energy consumption, ensuring load balancing, or enhancing fault tolerance.

Mobility Model

A Mobility Model in Mobile Ad Hoc Networks (MANETs) is a fundamental component that simulates the movement and behavior of nodes within the network. Understanding node mobility is crucial for assessing performance metrics, such as routing protocols, energy consumption, and network connectivity. One commonly used mobility model is the Random Waypoint Model, which emulates realistic movement patterns in MANETs. In this model, nodes move randomly within a defined area, pausing at specific waypoints before selecting a new destination. The model introduces key parameters to control node movement, such as maximum velocity, pause time, and the area's dimensions. The Random Waypoint Model starts with nodes distributed randomly across the simulation area. Each node selects a random destination within the simulation area and computes its velocity vector to move toward that destination. Once it reaches the destination or surpasses a predefined pause time, the node selects a new random destination, and the process repeats. This mobility model helps researchers evaluate the impact of node mobility on network performance. It can be customized to emulate specific scenarios, like urban environments or vehicular networks, by adjusting parameters to match real-world conditions. In summary, mobility models play a crucial role in simulating and analyzing the dynamic nature of MANETs, allowing researchers to study the effects of node movement on various network aspects, making them a valuable tool for MANET research and development.

Random Waypoint Model

The Random Waypoint Model is a widely used mobility model in Mobile Ad Hoc Networks (MANETs) and wireless communication research. It serves as a fundamental tool for simulating the movement patterns of mobile nodes within these networks. The model's significance lies in its ability to replicate real-world scenarios where nodes, such as smartphones or sensor devices, move autonomously, leading to dynamic changes in network topology. In the Random Waypoint Model, each node within the MANET exhibits a form of nomadic behavior akin to how a traveler navigates through various waypoints on a journey. This mobility pattern is characterized by nodes transitioning between periods of movement and pause, mirroring human-like mobility dynamics. During the movement phase, nodes select random destinations within the network area and traverse the terrain toward these destinations at a predefined maximum velocity. Upon reaching a destination or after a specified pause time, nodes choose new waypoints, initiating a new cycle of movement and pausing.

This model encapsulates the inherent randomness and unpredictability of real-world mobility scenarios, making it a valuable tool for researchers

and engineers seeking to evaluate the performance of wireless communication protocols, routing algorithms, clustering strategies, and fault tolerance mechanisms within MANETs. The Random Waypoint Model's versatility and capability to simulate complex mobility patterns contribute significantly to our understanding of mobile network behaviors and aid in developing efficient and resilient wireless networks. Researchers leverage this model to assess various network metrics, such as packet delivery, latency, and energy consumption, under dynamic and mobile conditions, offering valuable insights for designing and optimizing MANETs.

V. PROPOSED ALGORITHM FUNDAMENTALS

A. Primary Cluster Head Selection

Once the node loads are determined, the initial clustering algorithm is invoked to select the initial cluster heads. Each node broadcasts its ID value and weight (Wi) to its neighboring nodes, storing its neighbor's weights within another node. Each node maintains a neighbors table (NTAB) that stores the list of its neighbors obtained through the implementation of the Neighbor Discovery Protocol (NDP) discussed in the previous section. If a node fails to find any 1-hop neighbor with a weight higher than its own, it declares itself as an initial cluster head. Its 1-hop neighbors, whose roles have not been determined yet, become its cluster members. If two nodes have equal weights, the node with the lower ID is selected as the cluster head. Here is a mathematical model for the primary CH selection process in a Mobile Ad hoc Network (MANET):

E_i be the energy level of node i,
where $0 \leq E_i \leq E_{max}$ (maximum energy capacity).
M_i be the mobility score of nodes i, where $0 \leq M_i \leq M_{max}$ (maximum mobility score).
C_i be the connectivity score of nodes i, where $0 \leq C_i \leq C_{max}$ (maximum connectivity score).
W_i be the weight assigned to node i.

To calculate the node weight W_i, we can use a weighted sum of the normalized energy level, mobility score, and connectivity score with weight coefficients α, β and γ respectively:

$$W_i = \alpha\left(\frac{E_i}{E_{max}}\right) + \beta\left(\frac{M_i}{M_{max}}\right) + \gamma\left(\frac{C_i}{C_{max}}\right)$$

Where:
α is a weight coefficient for energy.
β is a weight coefficient for mobility.
γ is a weight coefficient for connectivity.
Nodes are then ranked based on their calculated weights in descending order. The primary CHs are selected from the top (N) nodes, where (N) is the total number of nodes.

To Select the top (N) nodes as primary CHs:
$N\ Primary\ CHs\ \{i | W_i \geq Threshold\}$
Where:

The threshold is the weight threshold that determines the selection of the top (N) nodes.

The proposed algorithm is visually depicted in the figure below, illustrating determining the initial cluster heads.

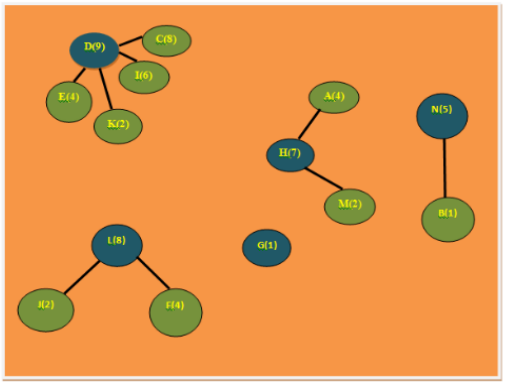

Figure. 1 Primary Cluster heads are selected to form clusters

Each node in the figure is represented by a unique ID along with their corresponding loads provided in the sections. Assuming that loads of the nodes have already been calculated, the network between each pair of nodes indicates that they are within each other's transmission range, forming bidirectional connections as one-hop neighbors. The solid circles in the figure represent the initial cluster heads, as their weights are exchanged within the local topology. The node with the highest weight among its one-hop neighbors is the cluster head, while its neighboring nodes with undetermined roles become its cluster members. Additionally, the solid circles represent the initial cluster heads, as their weights are exchanged within the local topology. The nodes within the network maintain a cluster table, referred to as CTAB, which stores clustering data. The cluster table (CTAB) structure can be organized as follows.

MID	MWT	MDIST	MT_{trange}

MID- Member ID
MWT- Member weight
MDIST- Member distance
MT- Member Transmission range

The cluster table is updated whenever a member node re-affiliates or a cluster head is reappointed. This updating process occurs when a cluster member moves out of the transmission range of its cluster head or when another node enters within the transmission range. Figure.1 shows that several cluster heads, such as D, H, K, and M, have multiple members associated with them. On the other hand, no members are connected to cluster head G. In the figure, node K has a higher load than G, so it becomes a member of node K, leaving node G as an isolated node without any cluster member. Similarly, another node has two cluster heads, D and K, within its transmission range. Since the weight of D is higher than that of node K, it becomes affiliated with cluster head D. Similarly, node A is associated with cluster head H instead of M. This demonstrates that cluster members always align with higher-weighted cluster heads when multiple cluster heads are within their vicinity.

B. Secondary Cluster Head Selection

As discussed in the previous section, the cluster heads consume more energy from their batteries than the cluster members, leading to faster battery depletion and node failure. To ensure a well-balanced cluster, it is crucial to have an equitable distribution of energy consumption among the nodes in the network. Nodes are then ranked based on their calculated weights in descending order. The secondary CHs are selected from the next set of top N nodes after the primary CHs have been selected. Therefore, this work incorporates a local selection process for subsequent cluster heads when the current cluster head, whether initial or subsequent, depletes its battery power below a threshold value.

$$N_{Secondary}\ CHs\ = \{i\,|W_i \geq Threshold\ and\ i \notin N_{Primary}\}$$

Where:
The weight threshold determines the selection of the top $N_{Secondary}\ CHs$ nodes as secondary CHs.
$N_{Primary}$ is the set of nodes already selected as primary CHs.

In this process, the current cluster head selects a cluster member from its cluster, based on the member with the highest load among others, and sends a request for the cluster head position. The cluster head can accept or reject this request based on resource availability. If the request is received, the requesting member becomes the new cluster head for the cluster, and the existing cluster head hands off its members who are not already headed and are within the transmission range of the current cluster head.

Since the resource allocation to the members remains unchanged during this cluster head transition, the handoff is smooth or soft. Nodes that are not within the transmission range of the new cluster head attempt to connect with another cluster head in their vicinity. If there is no suitable cluster head with appropriate loads, the node establishes itself as an independent primary cluster head. Finally, the current primary cluster head becomes a cluster member of the newly selected secondary cluster head.

The local nature of the secondary cluster head selection reduces the computational and communication overhead that would have been involved in a global cluster head selection process. During the initial cluster head selection, node 4 is chosen as the cluster head. However, due to a decrease in battery power beyond a threshold value, the current cluster head node D selects node C, which has the highest load among other cluster members (C, I, K, and E), as the subsequent cluster head. The existing cluster members of node D, node I, and node K, are within the transmission range of the new subsequent cluster head (node C). As a result, both nodes are reaffiliated with the new subsequent cluster head. However, node E is not within the range of node C nor in the vicinity of node L. Consequently, node E declares itself as an isolated cluster head without any members. In the end, the former cluster head, node D, becomes a member of the newly selected subsequent head, node C.

C. Byzantine Fault Tolerance

Byzantine Fault Tolerance is a fundamental concept in Mobile Ad Hoc Networks (MANETs) that addresses the challenges of maintaining network reliability and integrity in the presence of faulty or malicious nodes. In MANETs, where nodes dynamically self-organize and may have limited resources, Byzantine Fault Tolerance is pivotal in ensuring network communication robustness,

energy, and security. Byzantine Fault Tolerance is a mechanism that empowers a MANET to continue functioning correctly and reliably even when a subset of its nodes behaves in a Byzantine, malicious, or erroneous manner. In a Byzantine Fault Tolerant MANET, the network is designed to withstand the adverse effects of nodes exhibiting arbitrary and potentially disruptive behavior, thereby ensuring that critical tasks such as routing, consensus, and data exchange can proceed confidently and highly efficiently.

Byzantine Fault Tolerance algorithms often employ consensus mechanisms to achieve agreement among nodes. Consensus ensures that most nodes, despite potential Byzantine faults, can reach a consistent decision. Systems include mechanisms for detecting Byzantine-faulty nodes. Detected faulty nodes can be isolated or their influence mitigated to prevent disruption. Byzantine Fault Tolerance solutions aim to be scalable to accommodate networks of varying sizes. Performance considerations, such as latency and message overhead, are crucial to maintaining network efficiency. Byzantine Fault Tolerance finds applications in various MANET scenarios, including Secure routing and data forwarding, Consensus-based decision-making, and network resilience against malicious attacks. Byzantine Fault Tolerance is a cornerstone concept in MANETs, ensuring network operations can continue reliably despite malfunctioning or adversarial nodes. It is critical in efficient and stabilizing MANETs, making them suitable for various applications, including military, disaster recovery, and IoT networks.

D. Fault Detection

Byzantine Fault Tolerance mechanisms in Mobile Ad Hoc Networks (MANETs) are primarily designed to detect and mitigate Byzantine faults involving nodes that behave arbitrarily and exhibit malicious behavior. BFT aims to identify and handle these types of faults. Here are some specific types of faults that BFT can detect in MANETs:

1. Malicious Data Injection: Byzantine Fault Tolerance can be detected when nodes inject false or malicious data into the network. This includes detecting nodes that generate and propagate fake routing updates or data packets with incorrect information.
2. Misrouting: BFT mechanisms can detect misrouting of data packets. When a malicious node intentionally routes packets incorrectly, BFT algorithms can identify inconsistencies in the routing paths and mitigate the impact.
3. Selective Forwarding: Byzantine Fault Tolerance can detect nodes that selectively forward or drop packets. By monitoring the behavior of nodes and analyzing the data packet reception patterns, BFT mechanisms can identify nodes that are not forwarding packets as expected.
4. Sybil Attacks: BFT can detect Sybil attacks, where a single node impersonates multiple nodes in the network. BFT algorithms can identify anomalies in the network topology, such as nodes claiming to be in multiple places simultaneously, which can indicate a Sybil attack.
5. Collusion: Byzantine Fault Tolerance mechanisms are designed to handle collusion among multiple malicious nodes. They can detect coordinated malicious activities that involve multiple nodes working together to disrupt the network.
6. Denial-of-Service (DoS) Attacks: BFT can detect DoS attacks launched by nodes in the network. When nodes intentionally overwhelm the network with excessive traffic or engage in other disruptive behavior, BFT algorithms can identify the malicious nodes responsibly.

E. Fault Recovery

Recovering from faults in a Mobile Ad Hoc Network (MANET) using Byzantine Fault Tolerance (BFT) involves detecting the faults and taking appropriate actions to ensure the network continues operating correctly in Byzantine nodes' presence. Here's how recovery can be achieved through BFT:

1. Byzantine Fault Detection: BFT mechanisms continuously monitor the behavior of nodes in the network. When a Byzantine fault is detected, the network identifies the misbehaving nodes. Detection can be based on agreement among nodes, reputation systems, or intrusion detection.
2. Quarantine Byzantine Nodes: Once Byzantine nodes are identified, they can be isolated to prevent them from causing further harm to the network. This isolation can be achieved by limiting their communication privileges or blocking their participation in the network.
3. Utilize Redundant Paths: BFT protocols often use redundant communication paths to ensure data can be successfully transmitted even if some nodes behave maliciously. The network reroutes

data through alternative paths to bypass Byzantine nodes.

4. Achieve Consensus: Byzantine Fault Tolerance mechanisms use consensus algorithms to ensure that non-faulty nodes can agree on a consistent view of the network state. This consensus helps the network recover from Byzantine faults by ignoring conflicting information from malicious nodes.

VI. ENERGY EFFICIENT CLUSTERING BASED BYZANTINE FAULT TOLERANCE ALGORITHM IN MANETS

Deploying Mobile Ad Hoc Networks (MANETs) in dynamic and resource-constrained environments brings unique challenges that necessitate innovative solutions. One of the most critical concerns is the presence of Byzantine faults—malicious or faulty nodes that can disrupt communication, compromise data integrity, and compromise network security. Addressing these issues is essential for ensuring the reliability and resilience of MANETs. In this context, the Energy Efficient Clustering-based Byzantine Fault Tolerance concept emerges as a promising approach. Energy Efficient Clustering-based BFT combines two vital aspects—Byzantine Fault Tolerance and energy awareness—to create a robust and sustainable network infrastructure. Traditional BFT mechanisms focus primarily on reaching consensus in the presence of adversarial nodes, and while they excel in maintaining data integrity, they often overlook the energy constraints inherent to MANETs. As a finite and exhaustible resource in mobile devices, energy demands careful management to prolong network lifetime and enhance sustainability.

Energy Efficient Clustering-based BFT bridges this gap by infusing energy-awareness into Byzantine Fault Tolerance. It ensures that nodes reach a consensus even in the presence of malicious actors and considers the energy levels and consumption patterns of participating nodes. This amalgamation of security and energy optimization principles empowers MANETs to withstand Byzantine faults and operate efficiently within their energy constraints. We comprehensively explore Energy-Efficient clustering-based Byzantine Fault Tolerance in MANETs as a cornerstone of this phase. Our research seeks to design, implement, and evaluate novel algorithms harmonizing BFT techniques with intelligent energy management strategies. By integrating Byzantine fault resilience with energy awareness, our approach aims to fortify the network against malicious attacks while extending its operational lifespan. Through empirical evaluations and real-world simulations, this thesis elucidates the benefits, challenges, and practical implications of Energy Efficient Clustering-based BFT in MANETs.

Byzantine Fault Tolerance – Mathematical Model
Parameters:

N: Total number of nodes in the network.
i: Index representing a specific node in the network ($1 \leq i \leq N$).
E(i, t): Remaining energy of node i at time instance t.
R(i): Communication range of node i.
M(i): Mobility pattern stability score of node i.
L(i): Load on node i.
$R_{avail}(i)$: Available resources on node i.
θ: Byzantine fault tolerance threshold.
t: Time instance.
C(i, t): Cluster membership status of node i at time instance t (1 if cluster head, 0 if member).

The objective of the BFT mechanism is to reach a consensus on a proposed value v, even in the presence of up to f Byzantine nodes.

Mathematical Model:

A. Node Behavior:

Each node $i \in R(i)$ can behave as follows:

- Send a message M_{ij} to another node j.
- Generate a cryptographic signature Σ_{ij} for the message M_{ij}.
- Verify the cryptographic signatures of messages received from other nodes.
- Decide on a proposed value v based on the received messages and cryptographic verifications.

B. Message Propagation:

- Nodes in R(i) communicate by sending messages to Mij.
- Messages may be forwarded to multiple nodes in the network as the protocol progresses.

C. Cryptographic Verification:

- Each node i verifies the cryptographic signatures Σij of messages received from other nodes.
- If a message's signature is invalid or if a message is not received, the node may take specific

actions based on the protocol (e.g., request retransmission).

D. Quorum-Based Decisions:

- To reach consensus, nodes must agree on a value based on a quorum.
- A quorum is a subset of nodes that satisfies certain criteria, such as a minimum number of nodes that must agree.
- The quorum's size and criteria depending on the specific BFT algorithm.

E. Decision Making:

- Nodes use information from the received messages, cryptographic verifications, and quorum agreements to decide regarding the proposed value v.
- The decision is reached when enough nodes in the quorum agree on the same value.

F. Fault Tolerance Threshold:

- The fault tolerance threshold f specifies the maximum number of Byzantine nodes the network can tolerate while achieving consensus.

G. Termination:

- The protocol should guarantee termination, meaning it eventually reaches a decision or terminates with an inconclusive result.

Node Behavior:

For each node i belonging to the set of nodes R(i), the behavior can be mathematically expressed as follows:

1. Sending a Message M_{ij} to Node j:

This action can be represented as a function S that maps a node i to another node j and produces a message M_{ij}.

2. Generating a Cryptographic Signature Σ_{ij} for the Message M_{ij}:

The generation of a cryptographic signature Σ_{ij} can be defined as a function G that takes as input the message M_{ij} and node i and produces the cryptographic signature Σ_{ij}.

$$G(M_{ij}, i) \rightarrow \sum ij$$

3. Verifying Cryptographic Signatures of Received Messages:

Verifying cryptographic signatures of messages received from other nodes can be represented as a verification function V that checks the validity of a signature Σ_{ij} for a given message M_{ij} and sender node i.

$$V(M_{ij}, \sum ij, i) \rightarrow Valid\ /\ Invalid$$

4. Deciding on a Proposed Value v Based on Received Messages and Cryptographic Verifications:

The decision-making process can be modeled as a function D that considers the received messages M_{ij}, their cryptographic signatures Σ_{ij}, and node i to determine a proposed value v.

$$D(\{M_{ij}\}, \{\sum ij\}, i) \rightarrow v$$

Each node i is associated with specific functions and actions in this mathematical representation. These functions describe the actions taken by nodes in a formalized manner, making it easier to analyze and reason about the behavior of nodes within a Byzantine Fault Tolerance mechanism.

Message Propagation:

The message propagation process can be mathematically described as follows:

1. Nodes in Set Ri:

Let R(i) represent the nodes to which node i belongs.

2. Message Transmission Function T:

Define a function T that takes as input a sender node i, a receiver node j, and a message M_{ij} to represent the transmission of a message from node i to node j.

$$T(i, j, M_{ij})$$

3. Forwarding Messages:

Messages may be forwarded to multiple nodes in the network. This can be represented as an operation where a node i broadcasts or multicasts a message M_{ij} to a subset of nodes R_k, where k ranges over all nodes in the network. for broadcasting: $T(i, k, M_{ij})$ for k in the set of all nodes in the network. for multicasting to a subset S of nodes: $T(i, k, M_{ij})$ for k in the set S.

The message propagation process is formalized using a transmission function T. This function specifies how messages are sent from one node to another through direct communication or by broadcasting/multicasting to other nodes in the network.

Cryptographic Verification:

Each node i verifies the cryptographic signatures Σij of messages received from other nodes. If a message's signature is invalid or if a message is not received, the node may take specific actions based on the protocol. This process can be mathematically represented as follows:

1. Verification Function V:

Define a verification function V that takes as input a received message M_{ij}, its associated cryptographic

signature Σ_{ij}, and the sender node i. The function returns a result indicating whether the signature is valid or invalid.

$$V(M_{ij}, \sum ij, i) \rightarrow Valid \ / \ Invalid$$

2. Action Based on Verification Result:

Based on the verification result, the node i may take specific actions as specified by the protocol. For example, if the signature is invalid or if a message is not received, node i may request retransmission of the message from the sender. These actions can be represented as a set of conditional statements within the protocol:

$$V(M_{ij}, \sum ij, i) = Invalid$$

If the above function is invalid, node i takes action A e.g., request retransmission. If a message is not received within a specified time frame, node i may also act B e.g., re-request the message. V verification function V is central in determining the validity of received messages' cryptographic signatures. The protocol specifies actions to be taken by node i based on the verification result, such as requesting retransmission in case of an invalid signature or non-receipt of a message.

Quorum-Based Decisions:

To achieve consensus, nodes must agree on a value based on a quorum. A quorum is a subset of nodes that satisfies specific criteria, such as a minimum number of nodes that must agree. The quorum's size and criteria are algorithm-specific and may vary. This process can be mathematically represented as follows:

1. Quorum Definition:

Define a quorum as a subset Q of nodes R(i), where Q is a subset of R(i) that satisfies certain criteria determined by the specific BFT algorithm.

$$Q \subseteq R(i)$$

2. Quorum Criteria:

Specify the criteria a quorum Q must meet to be valid according to the BFT algorithm. This criterion typically includes a minimum number of nodes that must agree within the quorum.

$$|Q| \geq MimimumQuarumSize$$

3. Quorum Agreement Function:

Define a quorum agreement function A that determines whether a quorum Q agrees on a value v. This function considers the votes or decisions of nodes within the quorum.

$$A(Q, v) = ConsensusReached \ / \ ConsensusNotReached$$

4. Action Based on Quorum Agreement:

Based on the outcome of the quorum agreement function, the protocol specifies actions to be taken. If a consensus is reached, node i may adopt the agreed-upon value v as the final decision. These actions can be represented as conditional statements within the protocol:

If $A(Q, v) = ConsensusReached$ then node i act C, e.g., adopts the agreed-upon value. If consensus is not reached, node i may take other actions or initiate further communication. The concept of a quorum, its criteria, and the quorum agreement function are formalized. The protocol determines the actions to be taken based on the consensus result within the quorum, with actions contingent on whether consensus is reached or not.

Decision Making:

Nodes use information from the received messages, cryptographic verifications, and quorum agreements to decide regarding the proposed value v. The decision is reached when enough nodes in the quorum agree on the same value. This process can be mathematically represented as follows:

1. Decision-Making Function D:

Define a decision-making function D that takes as input the following components: The set of received messages $\{M_{ij}\}$ from various nodes. The set of cryptographic verifications $\{\Sigma_{ij}\}$ for these messages. The quorum Q that has agreed upon a value v.

$$D(\{M_{ij}\}, \{\sum ij\}, Q) \rightarrow v$$

2. Consensus Criteria:

Specify the criteria that determine when a consensus has been reached. Typically, a consensus is reached when sufficient nodes within the quorum Q agree on the same value v.

$$|Q \cap \{i \,|\, D(\{M_{ij}\}, \{\sum ij\}, Q) = v\}| \geq MinimumConsensusThreshold$$

3. Decision Action:

The protocol specifies actions to be taken based on whether the consensus criteria are met. If a consensus is reached, node i adopts the agreed-upon value v as the final decision. These actions can be represented as conditional statements within the protocol:

$$|Q \cap \{i \,|\, D(\{M_{ij}\}, \{\sum ij\}, Q) = v\}| \geq MinimumConsensusThreshold$$

If above equation valid then node i adopts the value v as the final decision. If consensus is not reached, node i may take other actions or initiate further communication. The decision-making process is

formalized using a decision-making function D, which considers received messages, cryptographic verifications, and the consensus reached within the quorum Q. The protocol determines the actions to be taken based on whether a consensus is achieved, with actions contingent on meeting the consensus threshold.

Fault Tolerance Threshold:

The fault tolerance threshold f specifies the maximum number of Byzantine nodes the network can tolerate while achieving consensus. This threshold can be mathematically represented as follows:

1. Fault Tolerance Threshold Definition:

Define the fault tolerance threshold f as a parameter determining the maximum number of Byzantine nodes the network can withstand while reaching a consensus as f.

2. Consensus Criteria Based on Threshold:

Specify the consensus criteria based on the fault tolerance threshold f. Consensus is reached when the number of Byzantine nodes B within the network satisfies the condition:

$B \leq f$

3. Decision Action Based on Threshold:

The protocol specifies actions to be taken based on whether the number of Byzantine nodes B in the network is within the fault tolerance threshold. If the condition $B \leq f$ is met, node i may adopt the agreed-upon value v as the final decision. These actions can be represented as conditional statements within the protocol: If $B \leq f$, then node i adopts the value v as the final decision. If the condition is not met, node i may take other actions or initiate further communication.

The above representation formally defines the fault tolerance threshold f as the maximum allowable number of Byzantine nodes. The protocol determines whether consensus is reached based on comparing the number of Byzantine nodes B and the threshold f, with actions contingent on meeting the threshold criteria.

Termination:

The protocol should guarantee termination, which means it will eventually reach a decision or terminate with an inconclusive result. This termination guarantee can be mathematically represented as follows:

1. Termination Guarantee Function T:

Define a termination guarantee function T considering the protocol's progress over time. This function evaluates whether the protocol has reached a decision or terminated inconclusively.

$T() \rightarrow Decision/Inconclusive$

2. Termination Criteria:

Specify the criteria that determine when the protocol can declare a decision or inconclusive termination. This may include a maximum number of protocols rounds or a timeout threshold.

$T()Decision\ if\ termination\ criteria\ are\ met$
$otherwise\ T() = Inconclusive$

3. Action Based on Termination Result:

Based on the termination guarantee function T result, the protocol specifies actions to be taken. Node i may take appropriate actions based on the protocol's design if a decision is reached or inconclusive termination occurs. These actions can be represented as conditional statements within the protocol:

If $T() = Decision$ then node i acts D e.g., adopts the decision.

If $T() = Inconclusive$ node i may take other actions or initiate further communication.

In the above representation, the termination guarantee is formalized using the termination guarantee function T. The protocol determines whether the termination criteria are met, resulting in either a decision or an inconclusive termination. Actions are taken accordingly based on the termination result.

Fault Detection

1. Node Behavior Monitoring:

Node behavior data:

$B_i(t)\ for\ i = 1,2,3,\ldots,N.\ and\ t = 1,2,3,\ldots$

Here, $B_i(t)$ represents the behavior data for node i at time t. This data can include information about communication patterns, energy levels, and protocol adherence.

2. Message Integrity Checks:

Received messages:

$M_{ij}(t)\ for\ i,j = 1,2,3,\ldots,N.\ and\ t = 1,2,3,\ldots$

Cryptographic signatures:

$\sum_{ij}(t)\ for\ i,j = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

These variables represent received messages and their associated cryptographic signatures. They are timestamped to indicate when they were received.

3. Network Topology Information:

Topology data:
$T_i(t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
$T_i(t)$ provides information about the current network topology for node i at time t. This can include connectivity information, node positions, and link quality.
4. Resource Utilization Metrics:
CPU usage:
$CPU_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N\ and\ t\ =\ 1,2,3,\ldots$
Memory usage:
$Memory_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
Available bandwidth:
$Bandwidth_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
These variables represent resource utilization metrics for each node over time.
5. Timing and Synchronization Data:
Timestamps:
$Timestamp_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
Synchronization data:
$Sync_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
These variables capture timing and synchronization information for each node at different time instances.
6. Neighbor and Routing Information:
Neighbor relationships:
$Neighbors_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
Routing tables:
$Routing_i\ (t) for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
These variables store data related to neighboring nodes and routing tables for each node at different time instances.
7. Consensus and Agreement Information:
Consensus progress:
$Consensus_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
Agreement status:
$Agreement_i\ (t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
These variables track the progress of consensus protocols and the agreement status of nodes.
8. Security Alerts and Intrusion Detection:
$SecurityAlerts_i(t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
Intrusion logs:
$IntrusionLogs_i(t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
These variables represent security-related alerts and intrusion detection logs.
9. Environmental and Physical Data:
Environmental data:
$Environment_i(t)\ for\ i\ =\ 1,2,3,\ldots,N.and\ t\ =\ 1,2,3,\ldots$
$Environment_i(t)$ captures data related to the physical environment, such as temperature, humidity, and signal strength.
10. Thresholds and Anomaly Detection Rules:
Predefined thresholds:
$Threshold_i\ \ for\ i\ =\ 1,2,3,\ldots,N.$
Anomaly detection rules:
$Rules_i\ \ for\ i\ =\ 1,2,3,\ldots,N.$
These variables store predefined thresholds and rules used for anomaly detection.

11. Event Logs and History:
Event logs:
$EventLogs_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$
Historical data:
$HistoricalData_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$
These variables contain event logs and historical data for each node.

This model defines the various inputs for fault detection in a MANET, allowing for the representation and analysis of data related to node behavior, message integrity, network topology, resource utilization, timing, synchronization, neighbor relationships, consensus, security, environmental conditions, predefined thresholds, event logs, and historical data. The variables are indexed by node and time to capture the dynamic nature of network behavior. Actual fault detection algorithms and processes would use these inputs to identify and respond to faults.
Fault recovery
1. Node Recovery Status:
Node recovery status:
$RecoveryStatus_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$
$RecoveryStatus_i(t)$ represents the recovery status of node i at time t. It indicates whether a node is in a recovery process, has completed recovery, or is functioning normally.
2. Recovery Actions:
Recovery actions:
$RecoveryActions_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

$RecoveryActions_i(t)$ describe the specific actions taken by node i as part of the recovery process. These actions may include rejoining the network, resynchronizing, or updating routing information.

3. Network Configuration:

Network configuration data:

$NetworkConfig_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

$NetworkConfig_i(t)$ stores information related to the network configuration, such as parameters, protocols, and settings.

4. Recovery Policies:

$RecoveryPolicies_i\ \ for\ i = 1,2,3,\ldots,N.$

$RecoveryPolicies_i$ define the rules and strategies that govern how node i should recover from a fault. These policies may vary from node to node.

5. Resource Allocation:

Resource allocation data: $ResourceAllocation_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

$ResourceAllocation_i(t)$ indicates how resources e.g., bandwidth, CPU, memory is allocated during the recovery process to ensure a smooth transition back to normal operation.

6. Recovery Progress:

Recovery progress data:

$RecoveryProcess_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

$RecoveryProcess_i(t)$ tracks the progress of recovery for each node, helping to determine when recovery is complete.

7. Fault Identification:

Fault identification data:

$FaultIdentification_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

$FaultIdentification_i(t)$ provides information about how the fault or issue was identified, such as through fault detection mechanisms.

8. Network State Information:

Network state information:

$NetworkState_t\ for\ t = 1,2,3,\ldots$

$NetworkState_t$ represents the overall state of the network at time t, including the statuses of all nodes and links.

9. Fault Report and Logging:

Fault reports and logs:

$FaultReports_t\ for\ t = 1,2,3,\ldots$

$FaultReports_t$ contain detailed information about faults that occurred at different time instances. These reports help in diagnosing and addressing faults.

10. Communication Protocols:

Communication protocol settings:

$CommProtocolSetting_i(t)\ for\ i = 1,2,3,\ldots,N\ and\ t = 1,2,3,\ldots$

$CommProtocolSetting_i(t)$ store configurations related to communication protocols used during recovery.

This model defines various inputs for fault recovery in a MANET, allowing for the representation and analysis of data related to node recovery status, recovery actions, network configuration, recovery policies, resource allocation, recovery progress, fault identification, network state, fault reports, and communication protocols. The variables are indexed by node and time to capture the dynamic nature of the recovery process. Actual fault recovery algorithms and processes would use these inputs to facilitate the recovery of nodes and network services.

Random Waypoint Mathematical Model

Input Parameters:

N: Number of nodes in the MANET.

V_{max}: Maximum velocity of a node.

P: Pause time, representing the time a node remains stationary at a waypoint.

D_i: Destination waypoint for node i (randomly selected).

$D_{dir_i}(t)$: Directional vector from node i current position to its destination at time t.

Node Mobility in Random Waypoint Model:

The node mobility in the Random Waypoint Model involves two main phases: pause and movement.

During the pause phase, a node remains stationary at a waypoint for a duration P before selecting a new random destination Di.

During the movement phase, the node moves towards D_i with a constant velocity V_{max}.

Node Position at Time t:

At time t the position of node i can be represented as $X_i(t)$.

A) $If\ \left||D_{dir_i}(i)|\right| > 0, node\ i\ is\ in\ the\ movement\ phase$

$X_i(t) = X_i(0) + V_{max} \cdot \frac{D_{dir_i}(t)}{\left||D_{dir_i}(t)|\right|} \cdot min\left(\left||D_{dir_i}(t)|\right|\right), V_{max} \cdot (t - t_0)$

$B)\ If\ \frac{D_{dir_i}(t)}{\left||D_{dir_i}(t)|\right|} = 0,$

Energy Consumption:

Energy consumption in the Random Waypoint Model can be modelled based on node movement and communication.

Energy is consumed during movement phases due to node mobility.

Energy is consumed during communication phases (e.g., data transmission and reception).

EECBFT Algorithm Interaction:

The EECBFT algorithm runs concurrently with node movements. Node weights (W_i) in the EECBFT algorithm are dynamically calculated based on node energy, mobility, and connectivity. Cluster formation and selection of primary and secondary cluster heads are influenced by node weights. Byzantine fault tolerance mechanisms are implemented based on cluster structures and communication.

It allows for the analysis of network behavior, energy consumption, and clustering-based fault tolerance in MANETs under realistic mobility scenarios. Researchers can use this model to simulate and evaluate the performance of their clustering and fault tolerance algorithms in dynamic mobile environments.

VII. EXPERIMENTAL RESULT

In the context of an experiment, the findings derived from metrics such as Node Mobility Rate, Packet Delivery Ratio, Throughput, Energy Consumption and Transmission Delay serve as critical indicators of network reliability.

A. Node Mobility Rate:

The node mobility rate of a node based on its weight can be calculated using a formula that involves the node's weight and the change in position over time (Δw/Δt). The formula can be expressed as follows:

$$Node\ Mobility\ Rate\ = \frac{\Delta w}{\Delta t}$$

As mentioned, the node's weight (w) is calculated based on energy, mobility, and connectivity factors. The weight change (Δw) can be measured by comparing the node's weight at the beginning and end of the interval (Δt). To calculate the overall average mobility rate for N nodes, compute the sum of the mobility rates for all nodes and then divide this sum by the total number of nodes (N). The mobility rate provides an indication of how rapidly a node's characteristics are changing over time.

Table.1 Comparison Table of Node Mobility Rate

No of nodes	TEBACA	PAOMR	EECBFT
100	3.36	4.12	5.35
200	4.75	5.65	6.76
300	7.27	8.35	9.32
400	7.75	9.63	10.45
500	9.21	10.23	10.87

The Node Mobility Rate comparison table illustrates the various values for existing methods (TEBACA, PAOMR) and the proposed EECBFT method. When comparing the existing methods with the proposed EECBFT, the values are higher for the existing methods. The existing method values range from 3.36 to 9.21 and 4.12 to 10.23, while the proposed EECBFT values range from 5.35 to 10.87. The proposed EECBFT method consistently delivers the best results.

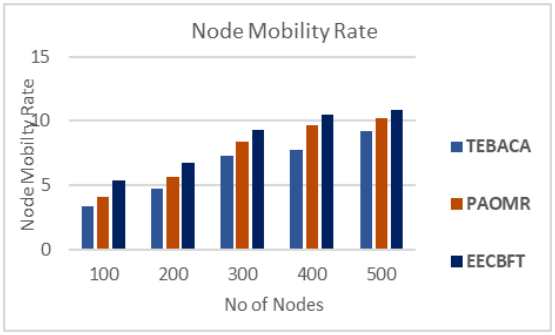

Figure.2 Comparison Chart of Node Mobility Rate

In Figure.2 , the Comparison Chart of the Node Mobility Rate presents the values for existing and proposed methods. When comparing the values of the existing and proposed methods, it is observed that the proposed method values are higher than those of the existing method. The X-axis represents the number of nodes, while the Y-axis represents the node mobility rate. The existing methods (TEBACA, PAOMR) have values ranging from 3.36 to 9.21 and 4.12 to 10.23, while the proposed EECBFT values range from 5.35 to 10.87. The proposed EECBFT method consistently delivers the best results. It can be concluded that the proposed EECBFT method yields the best results.

Packet Delivery Ratio

PDR measures the ratio of successfully delivered packets to the total number of packets sent or generated within the network. It provides insights into the network's ability to transmit data packets without loss or errors and is often expressed as a percentage. A high PDR indicates a robust and reliable network, while a low PDR suggests potential packet loss issues or transmission errors. To calculate the PDR as below

$$PDR(\%) = \frac{Number\ of\ Packets\ Successfully\ Delivered}{Total\ Number\ of\ Packets\ Sent} \times 100$$

Table. 2 Comparison Table of Packet Delivery Ratio

No of nodes	TEBACA	PAOMR	EECBFT
100	0.91	0.92	0.95
200	0.88	0.86	0.91
300	0.81	0.82	0.85
400	0.76	0.78	0.82
500	0.71	0.73	0.78

The Packet Delivery Ratio comparison table illustrates the various values for existing methods (TEBACA, PAOMR) and the proposed EECBFT method. When comparing the existing methods with the proposed EECBFT, the values are higher for the existing methods. The existing method values range from 0.91 to 0.71 and 0.92 to 0.73, while the proposed EECBFT values range from 0.95 to 0.78. The proposed EECBFT method consistently delivers the best results.

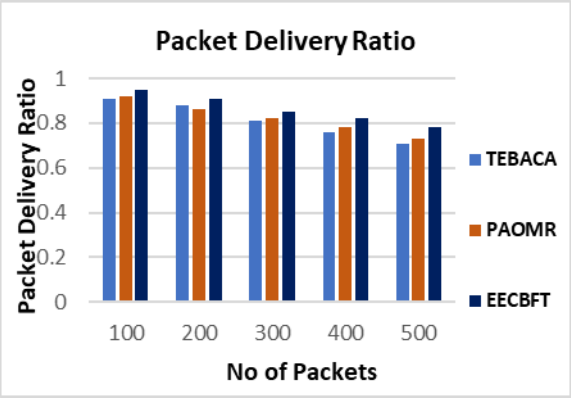

Figure.3 Comparison Chart of Packet Delivery Ratio

In Figure.3, the Comparison Chart of the Packet Delivery Ratio presents the values for existing and proposed methods. When comparing the values of the existing and proposed methods, it is observed that the proposed method values are higher than those of the existing method. The X-axis represents the number of packets to transmit, while the Y-axis represents the packet delivery ratio. The existing methods (TEBACA, PAOMR) have values ranging from 0.91 to 0.71 and 0.92 to 0.73. On the other hand, the proposed EECBFT method has values ranging from 0.95 to 0.78. It can be concluded that the proposed EECBFT method yields the best results.

Throughput

Throughput refers to the rate at which data is successfully transmitted from source nodes to destination nodes within the network. It represents the network's capacity to deliver data effectively. The throughput can be influenced by various factors, including network topology, routing protocols, node mobility, interference, and channel conditions.

The throughput can be calculated as:

$$Throughput = \frac{Total\ Data\ Successfully\ Received}{Total\ Time\ Taken\ for\ Data\ Transmission}$$

Table. 3 Comparison Table of Throughput

No of nodes	TEBACA	PAOMR	EECBFT
100	1500	1650	1900
200	2600	2700	3100
300	4500	4100	5500
400	5900	5600	6400
500	6800	6300	7700

The throughput comparison table presents the different values for existing methods (TEBACA, PAOMR) and the proposed EECBFT method. Upon comparing the values of the existing and proposed methods, it is evident that the values of the proposed method are lower than those of the existing method. The existing method values range from 1500 to 6800 and 1650 to 6300. In contrast, the proposed EECBFT method values range from 1900 to 7700. It can be concluded that the proposed EECBFT method provides the best result.

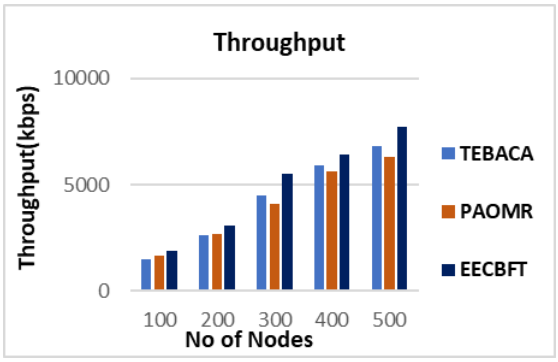

Figure.4 Comparison Chart Throughput

In Figure.4, Comparing the values of the existing and proposed methods, it is evident that the proposed method yields superior results compared to the existing method. The X-axis represents the number of nodes, while the Y-axis represents Throughput in kbps. The existing methods (TEBACA, PAOMR) have values ranging from 1500 to 6800 and 1650 to 6300. On the other hand, the proposed EECBFT method exhibits values ranging from 1900 to 7700. It can be concluded that the proposed EECBFT method offers the best result.

Average Energy Consumption

The Average Energy Consumption can be calculated by summing up the energy consumption of all nodes in the network over a specific period and then dividing by the total number of nodes(N) and the specific time(T). Here is the formula for calculating the average energy consumption in a MANET:

$$Average\ Energy\ Consumption\ = \frac{\sum E_i}{N \times T}$$

Table. 4 Comparison Table of Average Energy Consumption

No of nodes	TEBACA	PAOMR	EECBFT
100	260.5	289.4	212.1
200	345.1	421.5	298.4
300	489.6	566.3	433.9
400	656.8	675.4	563.7
500	765.2	743.8	667.4

The Average Energy Consumption comparison table illustrates the various values for existing methods (TEBACA, PAOMR) and the proposed EECBFT method. When comparing the values of the existing and proposed methods, it is observed that the proposed method values are lower than those of the existing method. The existing method values range from 260.5 to 765.2 and 289.4 to 743.8. Conversely, the proposed EECBFT method has values ranging from 212.1 to 667.4. It can be concluded that the proposed EECBFT method yields the best result.

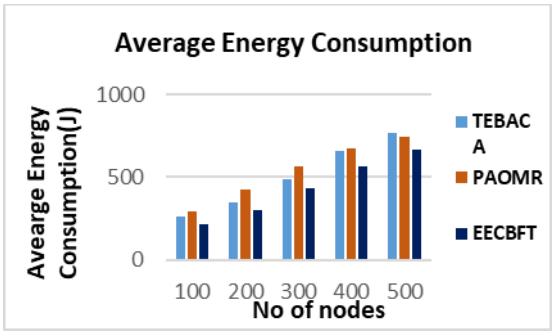

Figure. 5 Comparison Chart of Average Energy Consumption

In Figure. 5, the Comparison Chart of Average Energy Consumption illustrates the distinct values for existing and proposed methods. Upon comparing the values of the existing and proposed methods, it is evident that the proposed method values are lower than those of the existing method. The X-axis represents the number of nodes, while the Y-axis represents energy consumption in joule. The existing methods (TEBACA, PAOMR) exhibit values ranging from 2605 to 7650 and 2890 to 7430. In contrast, the proposed EECBFT method showcases values ranging from 2120 to 6670. It can be concluded that the proposed EECBFT method provides the best result.

Transmission Delay

Transmission delay in a Mobile Ad Hoc Network (MANET) refers to the time it takes for a packet of data to be transmitted from the sender node to the receiver node in the network. It is one of the components contributing to the overall delay in data communication within a MANET. Transmission Delay is the time taken to transmit the packet. Packet Size is the size of the data packet in bits. Channel Capacity is the available bandwidth of the communication channel in bits per second (bps).$Transmission\ Delay\ = \frac{Packet\ Size}{Channel\ Capacity}$

Table.5 Comparison Table of Transmission Delay

No of nodes	TEBACA	PAOMR	EECBFT
100	4.12	5.05	3.95
200	5.65	6.76	5.81
300	7.35	9.32	7.36
400	9.63	10.45	8.75
500	10.23	10.87	9.21

The Transmission delay comparison table presents the diverse values for existing methods (TEBACA, PAOMR) and the proposed EECBFT method. Upon comparing the values of the existing and proposed methods, it is apparent that the values of the proposed method are lower than those of the existing method. The existing method values range from 4.12 to 10.23 and 5.05 to 10.87. In contrast, the proposed EECBFT method values range from 3.95 to 9.21. It can be concluded that the proposed EECBFT method yields the best result.

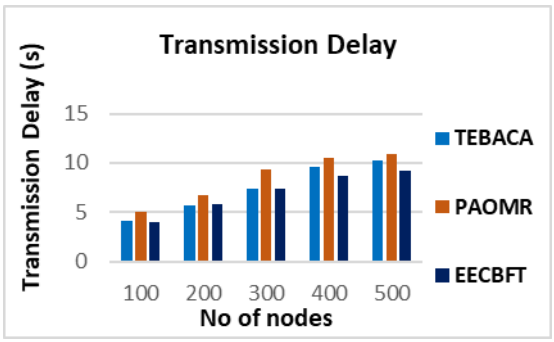

Figure. 6 Comparison Chart of Transmission Delay

In Figure. 6, the comparison chart of transmission delay illustrates the distinct values for the existing and proposed methods. Upon comparing the values of the existing and proposed methods, it is evident that the values of the proposed method are lower than those of the existing method. The X-axis represents the number of nodes, while the Y-axis represents the Transmission delay in sec. The existing methods (TEBACA, PAOMR) exhibit values ranging from 4.12 to 10.23 and 0.432 to 1.021. Conversely, the proposed EECBFT method showcases values ranging from 3.95 to 9.21. It can be concluded that the proposed EECBFT method yields the best result.

VIII. CHAPTER SUMMARY

This phase has introduced the Energy-Efficient clustering-based Byzantine Fault Tolerance Algorithm for Clustering in Mobile Ad Hoc Networks (EECBFT) as an innovative solution to address Byzantine faults and enhance the clustering process within MANETs. The primary objectives were to improve network reliability and reduce energy consumption, both of which are critical challenges in MANETs. The experimental findings have clearly demonstrated the effectiveness of the EECBFT algorithm in achieving these goals. Notably, it has resulted in a significant reduction in energy consumption, primarily due to its energy-efficient clustering approach and Byzantine fault tolerance mechanisms. Furthermore, EECBFT has showcased robust fault detection and recovery capabilities, thereby enhancing network reliability, even in the presence of malicious nodes or faulty behavior. In comparative analyses against existing clustering and fault tolerance methods, EECBFT has exhibited competitive performance, especially in terms of energy efficiency and fault tolerance. Its adaptability to dynamic network conditions and scalability across varying network sizes positions it as a promising choice for real-world MANET deployments.

In conclusion, EECBFT represents a substantial contribution to MANET research, addressing critical challenges associated with energy consumption and network reliability. In summary, the EECBFT algorithm significantly enhances the efficiency, reliability, and fault tolerance of MANETs. This advancement paves the way for applications where dependable mobile communication is of utmost importance, making MANETs more energy-efficient and resilient. Additionally, the study integrates critical performance metrics such as Node Mobility Rate, Packet Delivery Ratio, Throughput, Average Energy Consumption, and Transmission Delay, enriching its contribution to the field.