CHAPTER 4

PERFORMANCE IMPROVEMENT OF MOBILE AD HOC NETWORK USING PARTICLE SWARM OPTIMIZATION

4.1 INTRODUCTION

In MANETs, MAC layer misbehavior is an important research domain to improve the network performance. MANETs are gaining great popularity and getting deployed all over the world due to its dynamic nature and ease of deployment without any base station (Raptis et al 2009). It is a self organized network without depending on the pre-existing network infrastructure. It is an autonomous system, in which mobile nodes connected by wireless links are free to move randomly.

The system may operate in isolation, or may have gateways to interface with a fixed network. Here, each node acts as a router and provides multihop communication through the intermediate nodes (Guang and Assi 2005). The network topology may change rapidly and unpredictably over a period of time due to mobility. Mobile hosts can transmit messages directly to each other within their radio range, without any infrastructure devices (Chen et al 2008). The self-organizing features of rapid deployment make MANET very attractive in military applications and earthquake prone regions, where fixed infrastructure is not available (Wang & Teng 2013). In MANET, all the nodes have to cooperate to ensure a successful communication. However, the dynamic nature of MANETs is easily vulnerable to attack.
MAC protocol can be classified into two categories, namely, contention based and contention-free MAC protocols (Pornchaiwiwat and Benjapolakul 2006). IEEE 802.11 MAC protocol is an efficient protocol to share the wireless channel. Wireless networks started with ALOHA in the 1970s with distributed contention based protocol. Later, various protocols have been proposed in different papers like Multiple Access Collision Avoidance (MACA), MACA Wireless (MACAW). Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is the most popular contention based MAC protocol used in wireless networks.

In recent years, ad hoc networks mainly focused on achieving fairness and increasing spatial reuse through the distributed contention based shared algorithms (Luo et al 2001, Chao & Liao 2004, Cheng & Lu 2003).

Misbehavior is one of the major problems in MANET implementation. It may seriously degrade the performance of the network. It can be categorized into selfish (Aad et al 2004) and malicious misbehavior (Cardenas et al 2009). Selfish hosts characteristically misbehave to improve their own performance. This comprises of hosts that refuse to forward packets on behalf of other hosts with the intention to conserve energy. Another misbehaving technique of selfish nodes are to reduce their contention window by selecting a smaller contention window or using a different scheme instead of binary exponential backoff time.

Instead, malicious misbehavior aims mainly at disrupting the normal operation of the network (Guang & Assi 2005). This includes colluding adversaries that continuously send data to each other in order to deplete the channel capacity in their vicinity and hence prevent other legitimate users from communicating (Zhou et al 2004). Another example of malicious misbehaviors is the JellyFish (Aad et al 2004), which targets
closed-loop flows (such as TCP) that are responsive to network conditions (e.g., delays and loss).

Greedy hosts could exploit the vulnerabilities of IEEE 802.11 (Guang & Assi 2005) to enhance their share of bandwidth at the expense of other users. For example, IEEE 802.11 requires hosts competing for the channel to wait for backoff interval (Kim & Helmy 2010) before any transmissions. A selfish host may also choose to wait for a smaller backoff interval, in order to obtain an unfair advantage such as delay or higher throughput. By increasing their transmission probabilities, selfish nodes produce an increment in the number of collisions in the network, forcing the rest of the well behaved nodes to increase their backoff intervals, further increasing the advantage for the selfish nodes.

MAC layer uses a CSMA/CA mechanism to access the wireless channel. But it is easily vulnerable to different types of attack, due to the absence of centralized administrator. During communication, some nodes may violate the rules of CSMA/CA mechanism. These nodes are called malicious nodes. It uses a TO attack to disrupt the normal network operation. Here a malicious node chooses small or large SIFS value and forces the normal nodes for the timeout operation of the network. Hence, performance of the network is degraded, that is throughput, packet delivery ratio will be decreased and delay will be increased.

Optimization is a process that finds the best or optimal solution for a complex problem. PSO is a stochastic population based heuristic search technique which imitates finding food principle of bird swarm. In this work, a TO attack is presented to detect the malicious nodes using PSO algorithm (Valle et al 2008).
4.2 DETECTION METHODS

4.2.1 Trust Based Detection

Guang & Assi (2006) have proved that such selfish misbehavior can dangerously degrade the performance of the network and accordingly they proposed some modifications for the protocol by allowing the receiver to assign backoff values rather than the sender to detect and penalize misbehaving nodes. Raya et al (2006) have proposed a detection system called DOMINO, a system for detecting different selfish behaviors in WLANs. It detects nodes that do not hold on to the standard backoff mechanism. DOMINO cannot perfectly detect an attack where nodes do not freeze their back off counters due to ongoing transmissions. As a result, DOMINO cannot detect feasible CCA manipulations.

Selfish nodes intentionally misuse the MAC protocol rules to gain more access than well behaved nodes, so that they can try to save their battery power without forwarding the relaying messages (Toledo & Wang 2007). In addition, they do not intend to involve themselves in the network damaging activities. A selfish user may wait for a smaller back-off in order to obtain an unfair advantage such as delay or higher throughput. In case of malicious misbehavior, malicious nodes intend to disrupt the normal network operation like DoS attacks, TO attack or by choosing the smaller backoff value and by jamming the wireless channel to prevent communication. A misbehaving node can perform any type of misbehavior as long as it attains sufficient benefits, which causes a challenging problem to the design of countermeasures (Lu et al 2012).

4.2.2 Credit Based System

Recent research efforts have been focused on improving the security of ad hoc networks (Verotiana et al 2010, Nagel et al 2008, Kim and
Helmy 2010). Chao & Liao (2004) have proposed a credit based mechanism for assigning packet scheduling to achieve fairness. But the authors have not taken into account the problem of bottlenecks. Bottleneck nodes will degrade the performance of the network.

4.2.3 AD-MIX Protocol

Sundaramurthy & Royer (2003) employed the AD-MIX protocol to discourage the selfishness in MANET. This protocol tries to avoid the selfish misbehaving nodes in terms of forwarding data packets. Screening the true destination of a packet from intermediate nodes encourages data forwarding which forces a node to participate or risk the dropping of packets that may be destined for the node itself.

Pelechrinis et al (2009) has proposed a strong assumption that the selfish node has increased its Clear Channel Assessment (CCA) threshold to correctly recognize low power transmissions from the AP as legitimate packets. The AP can potentially detect such nodes by sending low power probes. This assumption involves that, if the power is lower than the CCA threshold, then the packet reception is not possible. This type of packet is treated as a noise. However, the attacker can avoid detection by simply changing the CCA threshold, only when it transmits a packet and while regressing back to the normal threshold right after the transmission. Also, packet reception success depends on how the radio transceiver is designed and may not be dependent on the CCA threshold. Also, this technique is not reactive.

4.2.4 Markov Chain Model

Bianchi (2000) has proposed a markov chain model to evaluate the performance of the DCF under an error free channel. The key assumption of
this model is that, at each transmission attempt regardless of the number of retransmissions suffered, the packet collides with constant probability. Chatzimisios (2003) developed a mathematical model to calculate the average packet delay. The transmission slot(s) time duration is equal to the average time duration of defer slots, which is because of the inaccuracy in each slot of this model.

Toledo and Wang (2007) calculated the sender backoff value indirectly from the inter arrival time based on the measurement of the channel idle period. Here the authors assumed that the nodes clock is synchronized and the channel activities of other nodes are perfectly observable.

**4.2.5 Cross Layer Mechanism**

There are several reports on the reduction of misbehavior in MANET. Kim & Helmy (2010) have proposed a Cross-Layer Attacker Traceback (CATCH) framework with a comprehensive set of protocol for reducing the false positive and false negative rates in MANETs. Jose & Srikant (2010) have proposed a mechanism called distributed and adaptive reputation mechanism for evading from revenge situations and to help restore cooperation quickly. Djenouri & Badache (2006) suggested an ACK mechanism and a bayesian approach to detect the packet dropping misbehavior in MANET, which monitors detects and isolates the misbehaving node. Yang et al (2005) has proposed an adaptive backoff window congestion control algorithm to maximize the total utilities with heterogeneous users.

Djahel & Nait-Abdesselam (2010) presented a novel solution to prevent and detect MAC layer misbehavior from the violation of the backoff computation rules and also to ensure a fair share of bandwidth among all the nodes. Giri & Jaggi (2010) have studied the various types of MAC layer misbehavior and have demonstrated the collective aggressive reaction
approach to ensure fairness in the network. Wu & Yu (2010) proposed a
threshold based method to detect the selfish node with the high detection rate
and less false detection rate. Choi et al (2011) have proposed a realistic way
of pinpointing the misbehaving nodes without requiring access of hardware
level information in IEEE 802.11 wireless LANs. They also introduced a
lightweight online decision algorithm using the sequential hypothesis test to
detect the misbehaving nodes.

Evolutionary algorithms such as PSO have been used recently in
several computer and networking applications (Huang et al 2009, Ali et al
2012, Dengiz et al 2011, Liu et al 2008). The main advantage of PSO is to
minimize the mathematical processing and global optimization capability
(Gajurel & Heiferling 2009). The objective of this work is to reduce the
number of misbehaving nodes in MANET using PSO algorithm.

4.3 MISBEHAVIOR IN MANET

Malicious nodes mainly disrupt the route discovery process and
forces packets to be rerouted through another optimal path in order to save
battery power. In addition, these types of malicious nodes access the
channel with less contention period in order to get higher throughput.
They may modify the timeout operation of the IEEE 802.11 protocol
(Kennedy & Eberhart 1995). The TO attack changes SIFS parameter
which does not follow the procedure of communication.

In order to manipulate the channel completely, a node could
transmit a signal after the short SIFS with an aim to achieve a notable
increase in the bandwidth. The sender transmits RTS after waiting for a
DIFS interval. The receiver node receives first RTS from the sender. Then,
the receiver sends CTS to the sender after waiting for SIFS interval. At the
same time, receiver calculates TO_DATA, a timeout interval during which
it expects to receive data from the sender. Here the sender is a misbehaving node, so it chooses a small SIFS (SIFS*) to calculate the small TO\_CTS (TO\_CTS*) value instead of TO\_CTS. Hence, the CTS do not arrive within the TO\_CTS period.

\[ \text{TO\_CTS} = \text{T}_{\text{RTS}} + 2 \times \text{MaxPropDelay} + \text{SIFS} + \text{T}_{\text{CTS}} \]  \hspace{1cm} (4.1)

where \( T_{\text{RTS}} \), \( T_{\text{CTS}} \) are the transmission time of RTS and CTS control packets respectively.

Eventually the sender drops the CTS frame due to the timeout period and at the same time receiver repeatedly receives only RTS packet instead of DATA packet. Subsequently, the sender again sends the second RTS frame to the same receiver. In this activity, the receiver maintains a counter to record the number of RTS messages from the same sender. When the second RTS arrives at node R during TO\_DATA, then the sender suspects the presence of a misbehaving node (Cagalj et al 2004).

4.4 OVERVIEW OF PARTICLE SWARM OPTIMIZATION

The PSO is a robust stochastic optimization technique based on the movement and intelligence of swarms. It uses the behavioral models of bird flocking and swimming of fishes in groups for solving hard optimization problems (Kennedy & Eberhart 1995). In PSO, each particle represents a potential solution within a search space. Basic PSO control parameters are very sensitive especially inertia weight, acceleration coefficient, velocity clamping and swarm size (Bergh & Engelbrecht 2006). Wrong initialization of these parameters may lead to divergent or
cyclic behavior.

The main advantage of PSO is to minimize the mathematical processing and good optimization capability (Shi et al 2011). PSO has attracted many significant attentions from the researchers due to good performance, low computational cost and easy implementation. The compensation of PSO are that it involves no evolution operators, such as crossover and mutation operators, and it does not involve the adjustment of too many free parameters. The PSO has been used in many applications like artificial neural networks, biological information, dynamic optimization, multi objective optimization, power system etc. The PSO and Genetic Algorithms (GA) have similar behavior in comparison with other types (El-Abd 2008).

PSO has two main component methodologies. First being is artificial life which includes bird flocking, fish schooling and swarming. The second being evolutionary computation. PSO maintains a swarm of candidate solutions referred to as particles. Its key concept is that, particles are flown hyper dimensional search space and each particle being accelerated towards the best optimum solution found by the particles neighborhood. It can be implemented in the form of computer codes. So it is computationally inexpensive in terms of memory requirements and speed (Shayeghi et al 2009).

It lies somewhere in between evolutionary programming and the genetic algorithms. As in, evolutionary computation paradigms, the concept of fitness is employed and candidate solutions to the problem are termed particles or sometimes individuals, each of which adjusts its flying based on the flying experiences of both itself and its companion. At the initial stage of PSO, initial population of particles with random positions and velocities is created. In consequent iterations, each particle adjusts its position and velocity by its own experience and other particles information.
The PSO algorithm includes the following steps:

- **Initialization**: Generation of a population of particles with random D-dimensional positions and velocities using a uniform probability distribution function.

- **Evaluation**: Computation of fitness value (objective function) for each particle.

- **Comparison**: For each particle, its fitness value is compared with its best fitness value obtained in previous iterations (the fitness value of Pbest) and if the new position had a better fitness value. Then Pbest would be replaced by the new position. Also, the fitness value of each particle is compared with the best fitness value of the group obtained until the previous iteration (the fitness value of gbest). If the particle had a better fitness value, gbest would be replaced by that particle’s position.

- **Convergence**: If a sufficiently good fitness value is achieved or a maximum number of iterations are reached then the algorithm could be stopped. Otherwise it will continue to updating step.

- **Updating**: Here calculation of new velocity and position of each particle is obtained from Equations. (4.2) and (4.3) and finally returns to evaluation step (Sadeghpour et al 2011).

Each particle is defined as a potential solution to the problem in a D dimensional space. The particle i is represented in a D dimensional space.

\[ X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D}) \]  \hspace{1cm} (4.2)
Each particle maintains a memory of its previous best position. The best previous position of the \(i^{th}\) particle can be represented as

\[
P_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,D})
\]  

and the velocity for the \(i^{th}\) particle is represented as

\[
V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,D})
\]  

The particle position with the highest fitness value for the entire run is called the global best. The global best particle among all the particles in the population is represented by

\[
P_g = (p_{g,1}, p_{g,2}, \ldots, p_{g,D})
\]

The velocity vector of every particle is adjusted based on the best solution of itself and its neighbors. The position of the velocity adjustment made by the particle’s previous best position is called the cognition component. The position of the velocity adjustments using the global best is called the social component.

The PSO update equations which is described below (Shi & Eberhart 1996) are

\[
v_{id}(t + 1) = \omega v_{id}(t) + c_1 * \text{rand()} * (p_{id}(t) - x_{id}(t)) + c_2 * \text{rand()} * (p_{gd}(t) - x_{gd}(t))
\]  

where \(c1\) and \(c2\) are two positive constants, \(\text{rand}()\) is a random function in the range \([0,1]\), and \(\omega\) is the inertia weight.

The velocity update Equation in (4.6) has three major components (Valle et al 2008).
• The first component is referred to as “inertia,” or “momentum,” or “habit.” It forms the tendency of the particle to prolong in the same direction it has been traveling. This component can be balanced by a constant as in the modified versions of PSO.

• The second component is a linear attraction towards the best position ever found by the given particle whose corresponding fitness value is called the particles best scaled by a random weight. This component is referred to as “memory,” or “self-knowledge,” or “nostalgia,” or “remembrance.”

• The third component of the velocity update equation is a linear attraction towards the best position found by any particle whose corresponding fitness value is called the global best, scaled by another random weight. This component is referred to as “cooperation,” or “social knowledge,” or “group knowledge,” or “shared information.

\[
x_{id}(t+1) = x_{id}(t) + v_{id}(t)
\]  

(4.7)

In general, the inertia weight is set according to the following Equation

\[
w = w_{\text{max}} - \left(\frac{(w_{\text{max}} - w_{\text{min}})}{\text{iter}_{\text{max}}}.\text{iter}\right)
\]  

(4.8)

Equation (4.6) is used to calculate the particle’s new velocity according to its previous velocity and the distances of its current position from its best own position and the group’s best position. Then the particle flies towards a new position according to Equation (4.7).
The performance of each particle is measured according to a predefined fitness function, which is related to the problem to be solved. The inertia weight $\omega$ is employed to control the impact of the previous history of velocities on the current velocity. This influences the trade-off between global and local exploration abilities of the "flying points". Inertia weight in the range $[0.9, 1.2]$ on an average has a better performance. A larger inertia weight $\omega$, facilitates global exploration while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight $\omega$ can provide a balance between global and local exploration abilities and thus require less iteration on an average to find the optimum.

### 4.4.1 PSO Algorithm

**Algorithm**

1. Initialize the swarm of particle $P(t)$, position $X_i(t)$ of each particle $P_i$. $P(t)$ is random within the hyperspace with $t = 0$.

2. Evaluate the performance $F(X_i(t))$ of each particle, using its current position $X_i(t)$

3. Compare the performance of each individual to its best performance.

   If $F(X_i(t)) < P_{id}$ then

   
   $P_{id} = F(X_i(t))$

   
   $P_i = X_i(t)$

4. Compare $F(X_i(t))$ to the global best particle with $P_{gd}$

   
   $F(X_i(t)) < P_{gd}$
\[ P_{gd} = F(X_i(t)) \]
\[ P_g = X_i(t) \]

\((5)\) Change the velocity vector of each particle

\[ v_{id}(t + 1) = \omega * v_{id}(t) + \eta_1 * \text{rand()} * (p_{id}(t) - x_{id}(t)) + \eta_2 * \text{rand()} * (p_{gd}(t) - x_{gd}(t)) \]

where the second term is the cognitive component and the last term is the social component.

\((6)\) Move each particle to a new position:

\[ x_{id}(t + 1) = x_{id}(t) + v_{id}(t) \]
\[ t = t + 1 \]

\((7)\) Repeat steps from \((2)\) till \((5)\) until convergence occurs.

where \(p_{id}\) and \(p_{gd}\) are the pbest and gbest respectively,

\[ t \] - current iteration,

\(\omega\) - inertia weight,

Suitable values for \(\omega\), \(\eta_1\) and \(\eta_2\) may be chosen depending on problem.

\(\eta_1\) and \(\eta_2\) are acceleration constants.

\(\text{rand}()\) - random function within the intervals of (0,1),

\(\omega\) - inertia weight,

\(i\) - 1, 2, ..., \(N\), where \(N\) is the swarm size and

\(D\) - total dimension of each particle.
PSO is used to handle continuous as well as discrete variable problems. It needs to be tuned by varying only few parameters. In addition, it is easy to implement in which few lines of code are essential for the implementation. GA needs mutation and cross over operations. However, PSO requires only two equations, which are presented in Equations (4.6) and (4.7). The parameter \( \omega \) called inertia weight, introduced in the PSO (Parrott & Li 2009), is used to balance the global and local exploration abilities.

At any time \( t \), the locations of User Nodes (UNt) are known. The decision problem is to relocate the mobile nodes from their current positions at time \( t \) to their optimized value at time \( t + H \) which minimizes the misbehaving nodes at the same time. In other words, the PSO algorithm searches for the best location \( (x_{it(t+H)}, y_{it(t+H)}) \) of each node \( i \) at time \( t + H \). Clearly, the distance where a misbehaving node can cover during the time span of \( H \) is limited. The relocation decision of node \( i \) at time \( t \) is represented in the polar coordinate system as \( (r_{it}, \theta_{it}) \), where \( r_{it} \) and \( \theta_{it} \) are the rotation angle (in radians, counter clockwise) from the x-axis and the distance that node \( i \) travels at direction \( r_{it} \) in \( H \) time units, respectively.

As described above, the PSO algorithm searches for the best value of SIFS, \( TO_{CTS} (x_{it(t+H)}, y_{it(t+H)}) \) of each node \( i \) at time \( t + H \). However, due to the dynamic nature of the problem, at time \( t + 1 \), which is the start of a new optimization cycle, new location information of the users becomes available and hence the optimization problem is solved again. Based on the deployment decision at time \( t \), the locations of the misbehaving nodes at time \( t + 1 \) are determined as follows:
\[ x_{i(t+H)} = x_{it} + \cos(r_{iit}) \min\{v_{\text{max}}, v^{*}_{iit}\} \quad \forall i \in \mathcal{N}_t \]

and

\[ y_{i(t+H)} = y_{it} + \sin(r_{iit}) \min\{v_{\text{max}}, v^{*}_{iit}\} \quad \forall i \in \mathcal{N}_t \]

Equation (4.9) means that each node \( i \) is deployed at time \( t \) with a commitment to be at location \( (x_{it(t+H)}, y_{it(t+H)}) \) at time \( (t + H) \). However, at the beginning of time period \( t + 1 \), the actual locations of the user nodes become available. Although the optimization is performed for \( H \) time steps ahead, the decision implementation is in the next time step. In other words, an agent \( i \) is allowed to travel a maximum of \( v_{\text{max}} \) distance at the direction of \( r^{*}_{iit} \). (Orhan et al 2011)

4.5 SIMULATION PROCEDURE

To test the performance of the PSO algorithm under dynamic scenarios, a simulation procedure was created to randomly generate the locations of user nodes. In the simulation procedure, each user node \( i \) is assigned to a random starting point \( (x_{i0}, y_{i0}) \) and a random destination point \( (x_{iT}, y_{iT}) \) which follows a path with random perturbations to its destination. At every step time, each user is assigned to a random velocity \( U(v_{\text{min}}, v_{\text{max}}) \) as shown in Figure 4.1(Orhan et al 2011).
The velocity limits for user nodes are $v_{\text{max}} = 0.05$ and $v_{\text{min}} = 0.02$ respectively. User nodes stop when they reach their destinations. The population size was 50 in all runs and the PSO algorithm was run for 10 iterations during each time step of the simulation. The population size and stopping criteria are user set parameters in meta-heuristic algorithms. In practice, these parameters are experimentally concluded based on the factors such as complexity of the search space, the number of decision variables, and CPU time limitations.

4.5.1 Simulation Setup

MATLAB 7.0 was used to test the performance of the network. The simulations are carried out using 50 mobile nodes, moving in an area of 1000m*1000m. The IEEE 802.11 MAC protocol is used here. The random waypoint is used as the mobility model and the nodes have a speed of 10 m/s where packet size is 512 bytes. The maximum transmission range is set to 250 m. The total simulation time is 100 s for each single run. Each data point is averaged over 10 runs. The performance of the PSO algorithm is evaluated.
and compared with PSO and without PSO algorithm. The simulation parameters are shown in Table 4.1.

Table 4.1 Simulation parameters for the evaluation of PSO algorithm

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of nodes</td>
<td>50</td>
</tr>
<tr>
<td>Simulation area</td>
<td>1000 * 1000 m²</td>
</tr>
<tr>
<td>SIFS</td>
<td>10 µs</td>
</tr>
<tr>
<td>DIFS</td>
<td>50 µs</td>
</tr>
<tr>
<td>Slot time</td>
<td>20 µs</td>
</tr>
<tr>
<td>CWmin</td>
<td>32</td>
</tr>
<tr>
<td>CWmax</td>
<td>1024</td>
</tr>
<tr>
<td>Mobility model</td>
<td>Random Way Point</td>
</tr>
<tr>
<td>Propagation delay</td>
<td>1 µs</td>
</tr>
<tr>
<td>Transmission range</td>
<td>250 m</td>
</tr>
<tr>
<td>Carrier sensing range</td>
<td>550 m</td>
</tr>
<tr>
<td>Traffic type</td>
<td>CBR</td>
</tr>
<tr>
<td>Channel bit rate</td>
<td>11 Mbps</td>
</tr>
<tr>
<td>Packet size</td>
<td>512 B/packet</td>
</tr>
</tbody>
</table>

4.6 PERFORMANCE ANALYSIS

The throughput, delay, packet delivery ratio, misbehavior percentage based on swarm size, inertia and iterations are measured.
The Figure 4.2 shows the throughput of MANET with and without PSO. It is seen that the throughput for MANET optimized with PSO is higher than the case of MANET without PSO. It is found that PSO increases throughput by nearly 50%. Results demonstrate that the PSO algorithm increases the throughput of MANET by decreasing the number of misbehaving nodes.
Figure 4.3 shows the variation of nodes as a function of delivery ratio. It is observed from the figure that the delivery ratio decreases on increasing the number of nodes. However, with the PSO, the decrease in delivery ratio is found to be less. For instance, if the number of nodes increases from 20 to 50, a marginal decrease of delivery ratio from 0.72 to 0.4 is observed, whereas with PSO the ratio decreases only from 0.8 to 0.75.

![Graph showing delivery ratio vs number of nodes](image)

**Figure 4.4 Effect of delay with the increasing number of nodes**

The Figure 4.4 shows the delay of MANET with and without PSO by increasing the number of nodes. It is observed that the delay increases with increasing the nodes in both cases. However, the increase in delay with PSO is very less when compared to without PSO. Hence, the performance of the network has improved with PSO.
Figure 4.5 Percentage misbehavior shown as a function of swarm size

Figure 4.5 shows the effect of swarm size on misbehavior percentage. It is observed that the misbehavior percentage reduces nonlinearly with increase in swarm size. Further, misbehavior percentage reaches a minimum value as the swarm size is increased in the range of 5 to 10. Since a large swarm size causes more computational complexity, it is inferred that the swarm size for this application can be selected in the range of 5 to 10.

Figure 4.6. Estimation time shown as a function of swarm size
Figure 4.7  Variation of estimation time and misbehavior percentage is shown as a function of swarm size

The variation of estimation time is shown as a function of swarm size in Figure 4.6. It is seen that the estimation time increases linearly for increase in swarm size in the range of 2 to 10. The variation of estimation time and misbehavior percentage is shown as a function of swarm size in Figure 4.7. It is observed that a swarm consisting of 5 particles yields lowest number of misbehaving nodes using a small estimation time. Figure 4.8 shows the variation of estimation time as a function of inertia. A nonlinear decrease in the estimation time is observed as the value of inertia is increased from 0.1 to 0.9.
Figure 4.8 Estimation time shown as a function of inertia

Figure 4.9 Estimation time and the percentage misbehavior shown as a function of inertia
Further, the estimation time and the percentage misbehavior as a function of inertia are shown in Figure 4.9. It is seen that the number of misbehaving nodes is high for inertia values in the range of 0.2 to 0.5. Hence, it appears that an inertia value of 0.9 yields lower misbehavior percentage in short period of time.

![Figure 4.9 Misbehavior Percentage vs Inertia](image)

**Figure 4.9** Effect of Inertia on the Misbehavior Percentage

It was found that the smaller inertia, the misbehavior percentage is very high (Figure 4.10). However, for the medium and large inertia, the misbehavior percentage was found to be zero. Wang & Teng (2013) reported that the performance of PSO is sensitive to the settings of inertia weight and acceleration coefficient value. The high convergence speed is the welcome feature for the original PSO.

![Figure 4.10 Effect of Inertia on Misbehavior Percentage](image)
The Figure 4.11 shows the effect of iterations in the misbehavior percentage. It is found that the misbehavior percentage decreased as the iteration increases. With a small increase in iterations, there is a drastic decrease in the misbehavior percentage. Shi et al (2011) reported that the performance of PSO has improved by reducing the number of iterations. Hence, the quality of solutions of PSO preponderates by reducing the iterations.
Figure 4.12 Estimation time shown as a function of iterations

When the number of iterations is increased the estimation time also gets increased.

Figure 4.13 Estimation time and the percentage misbehavior shown as a function of iterations
The estimation time and the percentage misbehavior are shown as a function of iterations which is given in Figure 4.13. It is seen that when the number of iteration increases from 2 to 10, misbehavior percentage decreases. While increasing the iterations, the estimation time also linearly increases. Misbehavior percentage drastically decreases with less iteration is inferred here.

4.7 DISCUSSION

Misbehavior at the MAC layer can lead to performance degradation in ad hoc networks. Handling MAC layer misbehavior is an important requirement in guaranteeing a significant throughput. MANETs are easily vulnerable to attacks. This chapter discusses about the potential countermeasures against MAC layer misbehaviors and also briefs about the broad research area MANET, its capabilities and limitations. MAC layer misbehavior detection and prevention is a proven technique to improve the security. The performance of MANET is analyzed in terms of throughput, packet delivery ratio, delay, misdetection ratio, correct detection ratio, etc. This chapter identifies and investigates some of the factors affecting performance such as IEEE 802.11 MAC protocol, CSMA/CA scheme, wait time scheduling, etc.

PSO algorithm chooses an optimal value to avoid the misbehaving nodes participating in the network. But due to mobility, some nodes misbehavior cannot be predicted. In the simulation, each node is assigned a random source and destination point. Nodes move towards the best solution and best neighbor solution. The use of optimal value in case of choosing large or small value than the specified value will prevent the node immediately.

The ability of the proposed method is to improve the network performance during the communication, improve the packet delivery ratio,
throughput and decreases the delay and misbehavior percentage is evaluated through the extensive simulation.

It is analyzed from this study that

- Reduces the number of misbehaving node participate in the network.
- Throughput is improved by 60% after implementing the PSO algorithm.
- Frame delivery ratio is improved by 40% after implementing the PSO algorithm.
- Delay is reduced by around 23% with the implementation of PSO.
- Misbehavior percentage is reduced considerably after the implementation of PSO algorithm.

The overall performance of the proposed method is found to be good after implementing the PSO algorithm.