Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 4519357, 14 pages
https://doi.org/10.1155/2017/4519357

Research Article

Mobile Ad Hoc Network Energy Cost Algorithm Based on Artificial Bee Colony

Mustafa Tareq,[1] Raed Alsaqour,[2] Maha Abdelhaq,[3] and Mueen Uddin[4]

[1]*School of Computer Science, Faculty of Information Science and Technology, National University of Malaysia, 43600 Bangi, Selangor, Malaysia*
[2]*College of Computing and Informatics, Saudi Electronic University, Jeddah 23442, Saudi Arabia*
[3]*College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 84428, Saudi Arabia*
[4]*Department of Information Systems, Faculty of Engineering, Effat University, Jeddah 22332, Saudi Arabia*

Correspondence should be addressed to Raed Alsaqour; raed.ftsm@gmail.com

Received 3 December 2016; Revised 13 April 2017; Accepted 4 May 2017; Published 17 August 2017

Academic Editor: Bernard Cousin

A mobile ad hoc network (MANET) is a collection of mobile nodes that dynamically form a temporary network without using any existing network infrastructure. MANET selects a path with minimal number of intermediate nodes to reach the destination node. As the distance between each node increases, the quantity of transmission power increases. The power level of nodes affects the simplicity with which a route is constituted between a couple of nodes. This study utilizes the swarm intelligence technique through the artificial bee colony (ABC) algorithm to optimize the energy consumption in a dynamic source routing (DSR) protocol in MANET. The proposed algorithm is called bee DSR (BEEDSR). The ABC algorithm is used to identify the optimal path from the source to the destination to overcome energy problems. The performance of the BEEDSR algorithm is compared with DSR and bee-inspired protocols (BeeIP). The comparison was conducted based on average energy consumption, average throughput, average end-to-end delay, routing overhead, and packet delivery ratio performance metrics, varying the node speed and packet size. The BEEDSR algorithm is superior in performance than other protocols in terms of energy conservation and delay degradation relating to node speed and packet size.

1. Introduction

Wireless networks are a main concern in the communications field in recent years [1]. These networks can be utilized in various fields of technology, such as in personal area networks, military, and industrial setting. Wireless networks possess valuable attributes, such as easy installation, cost efficiency, and reliability, leading to their wide range of applications [2, 3].

A mobile ad hoc network (MANET) is commonly used to enable communication when common communication infrastructure is unavailable. This type of network is used in many applications such as in military [4, 5]. Owing to its wide range of applications, MANETs have become an active research topic. Network node plays two roles: a router for data packets prepared for other nodes and producer and consumer of data packet flow [6]. Despite these functions, the limited battery life and mobility of the node pose two important challenges in MANET research. The wireless topology of MANET can be changed efficiently and rapidly [7]. The benefit of this network is it can be linked to a broad Internet scale. The node in MANET has two purposes: it can be routed by another node and used as a router for other nodes. Nodes in a MANET move randomly and freely and can leave and join any time. Owing to nodes' mobile nature, the topology of the network is dynamically changing. Suitable routing protocol is required for the network to adapt to the changes in topology [8, 9].

A serious issue in MANETs is the insufficient power for hand-held devices. This insufficiency hampers the function of packet-forwarding in a MANET environment. Many traditional routing protocols do not consider the energy consumption of an individual node. Instead, they select the path with minimum hop count, leading to high power consumption.

If a single intermediate node runs out of energy, then the whole communication is interrupted. Moreover, as a result of low battery power, such protocols lead to low implementation and low number of activities performed by the nodes. In turn, these conditions lead to the simultaneous execution of multiple functions, and extra energy is needed to efficiently deliver the packets.

All wireless nodes need a continuous power source to ensure node availability and effectiveness. Modern wireless communication research is shifting towards a bio-inspired algorithm to enhance power efficiency and reduce the cost of a MANET. One of the bio-inspired algorithms is the artificial bee colony (ABC). This paper proposed bee dynamic source routing (BEEDSR) algorithm, which is the integration of ABC algorithm with dynamic source routing (DSR) routing protocol. The BEEDSR selects the best and most energy- and cost-saving path between the source and destination.

The rest of this paper is organized as follows. In Section 2, we provide the background and related work. In Section 3, we present the proposed BEEDSR algorithm. In Section 4, we introduce the simulation settings. In Section 5, we introduce the performance metrics. In Section 6, we provide the simulation results and evaluation. Finally, in Section 7, we give the conclusion and offer possible directions of future work.

2. Background and Related Work

2.1. Dynamic Source Routing. Dynamic source routing (DSR) is efficient and ideal for routing in multihop wireless ad hoc mobile node networks [10]. A network can independently organize and configure itself based on DSR, and such network does not require network infrastructure, preadministration, or administration. To ensure that data packets are successfully delivered despite node movements in network situations, DSR affords highly reactive services. DSR is distinct from other protocols because it is capable of source routing, implying that the transmitter knows the overall hop-by-hop route to the destination. A node maintains route caches containing the familiar source routes. The node then updates entries in the route cache as it learns about the new routes. The two major phases of the protocol are route discovery and maintenance [11, 12].

2.1.1. Route Discovery. In DSR, the route discovery mechanism is applied to find a route from source to destination and to transfer data in the case that a destination route is unknown by a source node. When a source node aims to send a packet to a destination, it searches for its route cache to ensure if it already contains a route to the destination. If it finds an unexpired route to the destination then it uses this route to send the packet to the destination node. However, if the source node route does not exist, then the route discovery will broadcast a route request (RREQ) packet to all participating nodes within the network. Each intermediate node checks whether it knows a route to the destination; otherwise, it appends its address to the route record of the packet and forwards the packet to its neighbors [10].

The information of all the relaying nodes towards the destination is stored in the RREQ packet. The source node receives a route reply (RREP) message when the destination node receives the RREQ packets. The RREP carries a copy of the route information collected by the RREQ. This route information is then stored by the source node for future communication. The route discovery process is illustrated in Figure 1, where node "Y" is the source and node "I" is the destination. Two RREQs are sent with identification numbers, that is, "id = 1" and "id = 2," and the routes to the destination are recorded as route 1 "YAMI" and route 2 "YSANI."

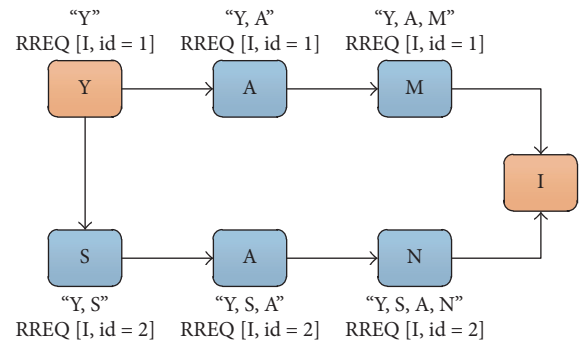

Figure 1: DSR route discovery mechanism [12, 13].

2.1.2. Route Maintenance. This mechanism is applied in the DSR during the packet communication from source to destination node. The DSR uses the route maintenance mechanism to search for known alternative routes which can be used to send packet from source to destination node [12, 13].

A broken communication link or a change in network topology can lead to failure in communication. As a result, a route error message (RERR) is immediately sent to the source node. After receiving a RERR message, the source node removes the hop in error from its route cache and starts a new route discovery. The nodes that forward the error packet along the way delete the entire route in the broken link from their own routing tables. The route discovery of the DSR protocol often discovers many routes from the source node to the destination node. The route with a minimal hop is also more likely to be chosen for data transmission than others are; in turn, the nodes that are frequently chosen are more likely to consume more energy, resulting in greater energy consumption and shorter battery life [10]. Route maintenance is illustrated in Figure 2, where node "Y" is the source and node "I" is the destination. When the link between nodes "M" and "I" fails, node "M" sends a RERR back to node "Y."

2.2. Artificial Bee Colony Algorithm. The ABC algorithm has recently attracted the attention of many researchers and academics for its ability to successfully solve combinatorial optimization [14]. ABC is inspired by the intelligent behavior, which refers to the acts of searching for food sources (known as nectar) by the honeybee.

In the ABC system, a bee colony can be classified as one of the three types based on the role it plays: the onlooker, the forager (employed), and the scout bees. Onlooker bees will

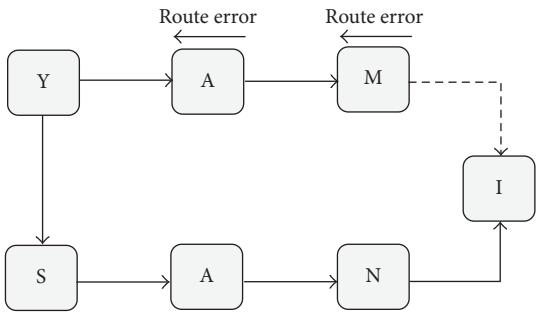

Figure 2: DSR route maintenance mechanism.

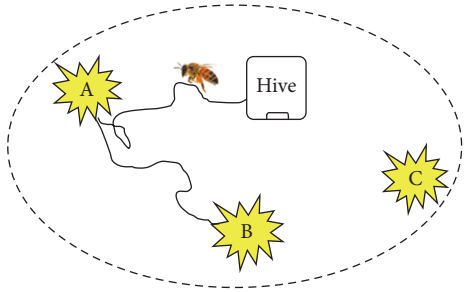

Figure 3: ABC scouting mechanism [14].

be waiting on the waggle dance area to make a decision on choosing a food source [15]. Forager bees are those that keep visiting their previously visited food sources to obtain nectar. Scout bees are those that conduct random searches to find new food sources. Figure 3 shows the process of the scout bee, wherein A, B, and C are the food sources, and the scout bee starts searching randomly from one source to another.

Initially, scout bees identify the positions of all food sources. Thereafter, both onlooker bees and forager bees exploit the nectar of food sources, and such continual exploitation ultimately exhausts food sources [16]. Afterward, the forager bee that had been exploiting the exhausted food source becomes a scout bee again in search for other food sources. The main steps of the ABC algorithm are as follows:

(i) Initialize population (food sources).

(ii) REPEAT:

(a) The employed bees visit the food sources based on their memory and evaluate the amount of nectar and return to the hive to do waggle dances in that location.

(b) Onlooker bees watch the waggle dance and go to the food sources based on their nectar amounts.

(c) The scout bees start their random search for new food sources when the food sources are exhausted. The best food source gets registered.

(iii) UNTIL (requirements are met).

Each cycle of searching for food sources depends on three steps. First, the initial food source locations are regenerated, which may be distributed randomly. Second, the forager bees are sent to collect the nectar from those food sources and measure the amount of nectar [15]. Those bees also share the information of food source when they return to their hive (i.e., distance, direction, and profitability) with the onlooker bees, which are waiting at the waggle dance area of the hive.

Third, the forager bees keep visiting the same food sources that are already stored in their memory, after which they check their neighborhood and update their information. Depending on the information on nectar amount distributed by all forager bees, onlooker bees decide on the best possible food source to visit so that they can exploit the nectar from it. Onlooker bees tend to choose a certain food source with much nectar. After a certain period passes, these food sources will be abandoned by the bees when the nectar becomes exhausted. Thereafter, a forager bee will become a scout bee and starts a new journey of discovering new food sources to replace the abandoned ones. At any point in time, only one bee will act as a scout bee, and it will be the one to determine a new food source. Such bees are called artificial scouts [17]. Figure 4 shows the ABC mechanism, in which two discovered food sources are assumed as A and B. Initially, a potential forager starts as an unemployed forager.

The forager bee is unaware of the food sources around the nest. For such bee, two scenarios are possible [17]:

(a) The bee begins to scout for food from one place to another near the nest because of internal or possible external motivation (scout).

(b) After watching the waggle dances, the bee can be a recruit and can begin to explore for a food source (foragers).

After finding a food source, a bee uses its own ability to memorize the site and begins exploiting the food source. In this way, the bee becomes an employed forager. Afterward, the bee brings the nectar to the hive and unloads it in the storage. After unloading, it has the following options:

(a) After leaving the food source, it may become an uncommitted follower (UF).

(b) Before returning to the food source, it may dance and recruit mates.

(c) Without recruiting any bees, it may continue to forage at the food source.

2.3. Related Work. In finding the path between the source and destination, most previous works deal with the problem of maintaining and finding correct route through changing topology and mobility. In [18], the authors examined the protocol based on swarm intelligence (SI). Furthermore, they proposed the bee-inspired protocol (BeeIP) to provide multipath routing in wireless ad hoc networks of mobile nodes.

The BeeIP uses foragers to collect path information and mark each path with a selection metric values on their way back. The importance of these metrics and their influence on the behavior of the protocol is very significant. A metric related to speed is used for the experimental comparison of BeeIP work, the summation of the transmission delay and

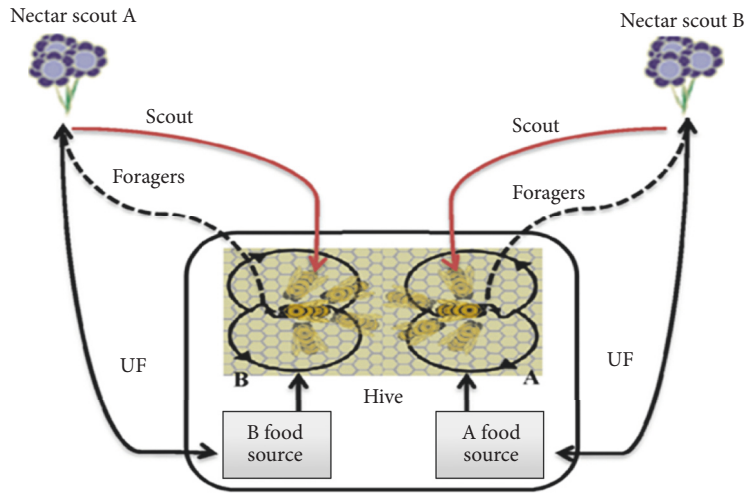

Figure 4: ABC food discovering mechanism.

queuing delay for each intermediate link of the path, from the destination towards the source. This ensures that the fastest path from the list is selected.

Since BeeIP is designed to evaluate routing based on path level, link breakage within a path is detected when no foragers return back to the source node within a period of time. The source node sets the path's foraging capacity to zero and marks the path as unacknowledged. The first ensures that no future foragers will be given the broken path's ID, whereas the latter allows the path to become available again, if a forager eventually comes back.

The simulation results indicate that BeeIP offers significant levels of adaptability and efficiency in its decision of which path is suitable to take. The findings showed that BeeIP generally has better performance than other protocols.

Sridhar et al. [19] proposed an insect colony-based intelligence, referred to as the SI technique. However, they proposed a sensor bee routing protocol for MANETs inspired by the foraging behaviors of honey bees. The results suggest that sensor bee delivers superior performance in latency, packet delivery, ratio, and less energy consumption compared with other SI algorithms.

To improve the energy aware reliable routing protocol (EARRP) in MANETs, an enhancement and design of such protocol have been suggested in [20]. Two SI techniques are involved in the protocol: bee colony foraging behavior and ant colony optimization (ACO). The authors compared EARRP with ad hoc on-demand distance vector (AODV) protocol using the following performance metrics: overhead, packet delivery ratio, total energy consumed by nodes, and delay. The results showed that EARRP is superior in performance than AODV in terms of overhead, delay, reduced energy consumption, and enhanced packet delivery ratio.

A honeybee algorithm has been suggested for ad hoc routing in [21]. The honey bee algorithm is a type of SI technique that is applied for optimization problems. In [21], the algorithm restructured the bee colony algorithm from the phase of initialization to the phase of execution and obtained better results than the current ABC methodology. In such aspects as time of response and throughput, the new system provides optimal values because the bee colony requests less running overhead. In turn, this leads to lower energy and less traffic, thus improving battery life and network efficiency.

In [22], the authors proposed predicted energy-efficient bee-inspired routing (PEEBR) algorithm. The algorithm was inspired by the bees' foraging behaviors and was compared with two state-of-the-art ad hoc routing protocols, namely, the destination sequenced distance vector (DSDV) and AODV, for different network MANET sizes [22]. They also compared the energy consumption of PEEBR, measured in mJ/KB, with another bee-inspired routing protocol called BeeAdHoc. The simulation results show that PEEBR is a competitive energy-efficient routing algorithm.

3. Proposed BEEDSR Algorithm

In this research, the ABC algorithm is combined with DSR routing algorithm to implement the BEEDSR routing algorithm. The foraging activities of bees are similar to the path activity of networks. In a MANET, all nodes should work cooperatively and efficiently by sharing information on the quality of the node links and partial routes. This process is similar to the food searching activity behaviors of a colony of bees. The ABC algorithm is suitable for dynamic, flexible, and multiobjective problems. The BEEDSR utilizes the ABC algorithm to search for the path through on-demand nature using context-aware metrics to select the best path. The DSR routing is used by the employed bees to locate possible paths.

The proposed BEEDSR algorithm can be described in the following steps:

(1) Nodes generation ($i = 1, \ldots, \mathrm{NN}$)

(2) Initialize the position of nodes

(3) Randomly initialize the speed of each node

(4) Select the source and destination node

(5) Initialize the population of solution X_i, $i = 1, \ldots, \mathrm{SN}$

(6) Each node broadcasts hello message to its neighbors' node to check the node in free or busy state

(7) For each particle X_i

Do

(a) While whole network is not covered

Do

(1) Select the route of X_i (the number of solution)
(2) Find the neighbors of the route
(3) Calculate the distance between each node

(8) Calculate nodes energy probability value P_i for the solution X_i using (1)

(9) End while

(10) End for

(11) Store the best energy routes in array (ID)

(12) While the maximum number of cycles is not reached

Do

(a) Select another route of X_i
(b) Calculate the probability value P_i for the solution step (8)
(c) Update the contents of (ID)
(d) Increment the loop counter
(e) When the current combination of routes of the solution is better than the combination of routes contained in its memory, the particle's position is updated

(13) End while

(14) Memorize the best solution achieved so far

(15) Broadcast the data from source to destination using DSR based on the best energy route.

In the first step, nodes ($i = 1$) of NN number of nodes (food source) are generated, where NN denotes the population size. Afterward, the population of solution X_i is initialized. Each solution X_i ($i = 1, 2, \ldots, \mathrm{SN}$) is a dimensional vector, where SN is the number of solutions. After initialization, each node broadcasts a message to its neighbor to check whether the node is free or in busy state.

Employed bees produce the position (solution) modification in its memory based on local information (visual information) by testing the energy amount (fitness value) of the new source (new solution). After the employed bees complete the search, they share both the energy and position information with onlooker bees. The onlooker bees evaluate the energy information from employed bees and then select a food source with the probability (P_i) related to the energy amount.

$$P_i = \frac{\mathrm{fit}_i}{\sum_{n=1}^{\mathrm{SN}} \mathrm{fit}_{i_n}}, \quad (1)$$

where P_i is the nodes' energy probability value through the route X_i, fit_i is the fitness value of route X_i, and SN is the total number of routes.

Identifying the number of nodes that will be fully power aware or have a power efficiency leakage is significant. Generating several nodes using node generator tool at NS2 is necessary to provide the scout bee with reasonable DSR nodes. Afterward, a swarm population will start to evaluate the nodes by using the evaluation problems.

Each time ABC visits a node (i.e., network topology or several nodes itself), it calculates the energy probability. The ABC consists of an initialization procedure and a main search cycle which is iterated until a solution of acceptable fitness is found for the best DSR-aware node. When onlooker bee arrives, the algorithm rigorously checked to examine if the area has a source route and the number of nodes present. If no source route is present, then it broadcasts the packet and rescan the energy for onlooker bee; it examines if it is an outgoing or incoming broadcast packet for transmitting energy.

In BEEDSR, the ABC mechanism integrated with the DSR, in terms of enhancing the routing process, selects the best route among the total nodes from the source to the destination. The scout bee also measures the power of both the node and node distance. If the node is allocated in a distant location, then it consumes a considerable amount of energy, resulting in greater energy loss.

Figure 5 shows the manner of specifying the source and destination nodes, and Figure 6 illustrates the initialization for the numbers of solutions using the BEEDSR algorithm.

To obtain the best result, a bee determines the distance between all nodes and their neighbors. Thereafter, each node sends or broadcasts a hello message to its neighbors to ensure that the node is in either a free or busy state. If the neighbor node replies, it means that the node is in a ready state. Otherwise, it is in a busy state and is not taking part in any other transmission. The path is selected based on the ready state nodes. Based on the bee algorithm, the source node sends the route request message to determine the best available path in the network, that is, the shortest path with minimum hop count and energy amount.

BEEDSR sends route request packet passes through the intermediate nodes. During this broadcast, ABC calculates the energy amount for each node based on fitness equation (1). ABC tries to figure out which better node has high power availability and leave the powerless nodes. In the final stage, ABC recommends the best path for DSR. Once the packet of route request reaches the destination, ABC selects the shortest path from source to destination based on the energy intermediate nodes with minimum nodes.

After the route request reaches the destination, it replies in ascending path. Based on that path, the data start transmitting packets through the source node. Figure 7 shows the bee searching process for the best available node with the shortest path. Figure 8 presents the optimal path selected.

4. Simulation Settings

BEEDSR algorithm simulation experiments are conducted using NS2. In these simulation experiments, a network setting

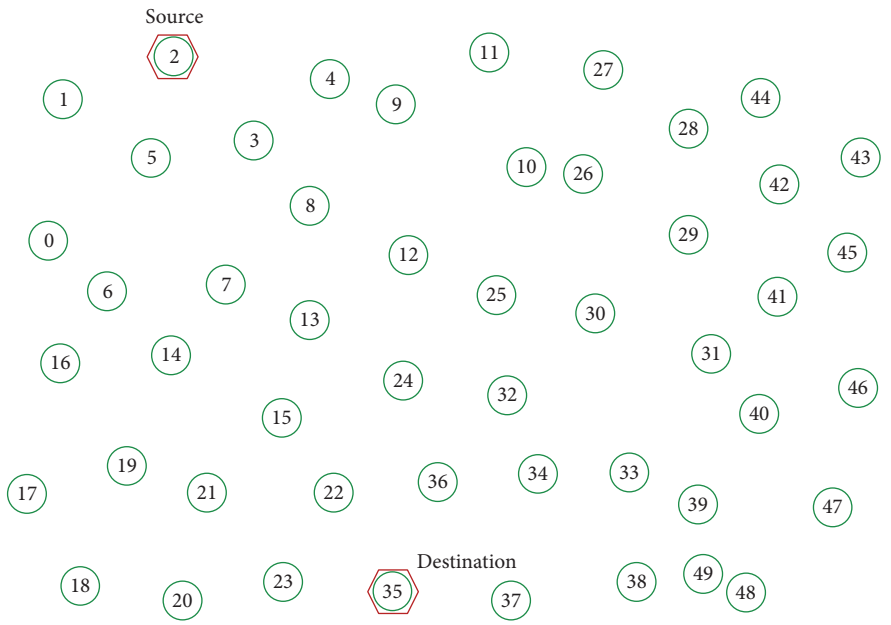

Figure 5: Specifying the source and destination in BEEDSR algorithm.

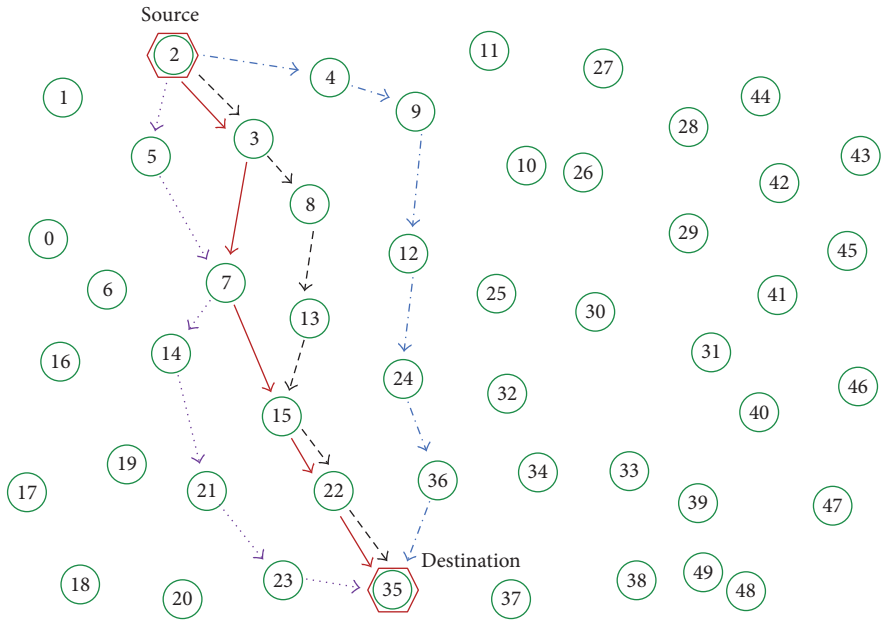

Figure 6: Initializing the numbers of solutions in BEEDSR algorithm.

on OTcl script has been defined, such as the routing protocol, propagation model, network traffic, and number of nodes to be used. The simulation produces two output files: a trace file used for data processing and a NAM file used to visualize the simulation. BEEDSR algorithm performance was compared with DSR and BeeIP algorithms. Figure 9 shows the steps of executing BEEDSR, BeeIP, and DSR on NS2.

Two different scenarios were selected to evaluate the performance of BEEDSR. In the first scenario, nodes are allowed to move within the maximum speed ranging 5–20 m/s. In the second scenario, the packet size has different sizes: 128, 256, 512, and 1,024 bytes. In the two scenarios, the constant bit rate (CBR) traffic sources were used, and the transmission range for each node was set to 250 m. The simulation area was set to 1,670 m × 970 m for all simulations, and the simulation time was set to 50 s. Fifty nodes were fairly distributed, and the initial energy was 100 joules. Two considerably distant nodes were selected to serve as the source (beehive) and the destination (flower). Nodes are moving randomly; thus, the network topology may change randomly and rapidly at unpredictable times. The data points presented in the simulation results were calculated as the average of 10 simulation

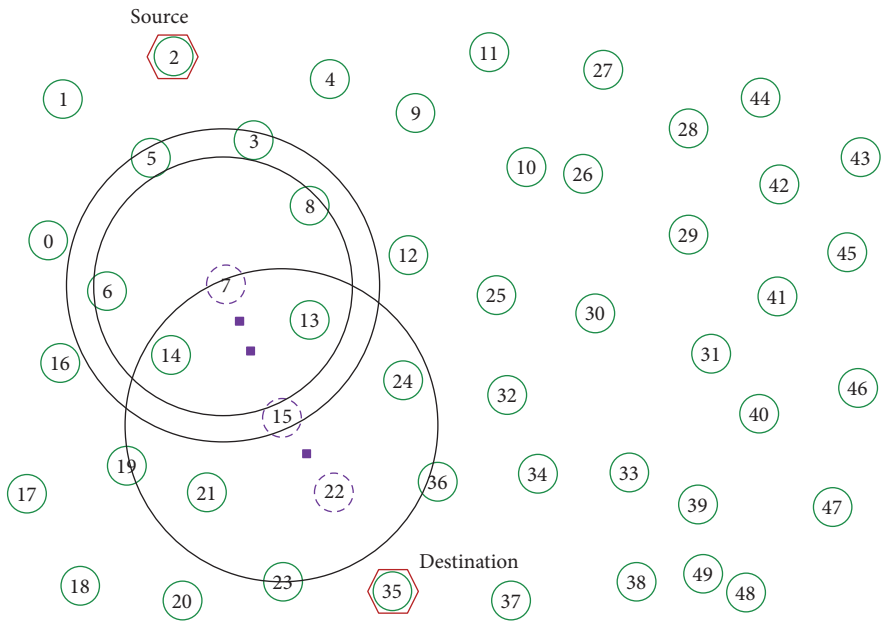

Figure 7: Search for the best available node and nearest path in BEEDSR algorithm.

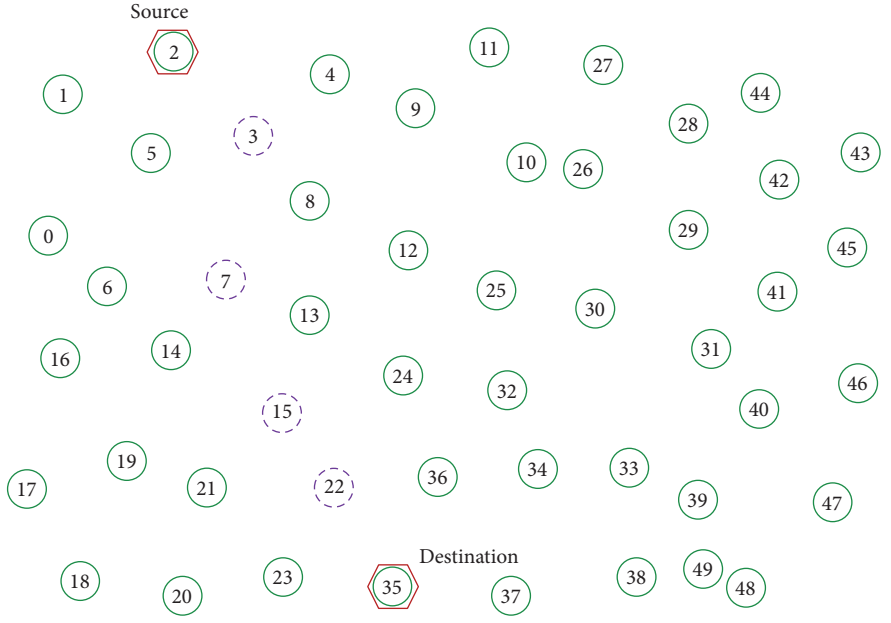

Figure 8: Path selection among the source to the destination in BEEDSR algorithm.

runs to eliminate the effect of any anomalous individual result because we observed a realistic variance among the points using 10 or more simulation runs. Table 1 summarizes the simulation parameters setting.

5. Performance Metrics

In this study, the aforementioned evaluation involves several performance metrics, which are discussed below.

5.1. Average End-to-End Delay. The average end-to-end delay metric is the average time it takes to broadcast the data packet successfully from the source to the destination through the network. This delay comprises several smaller delays in the network, including all possible delays caused by buffering during route discovery latency, queuing at the router interface queue, retransmission delays at MAC, propagation, and transmission time. The average E2E delay of data packets can be calculated using the following formula:

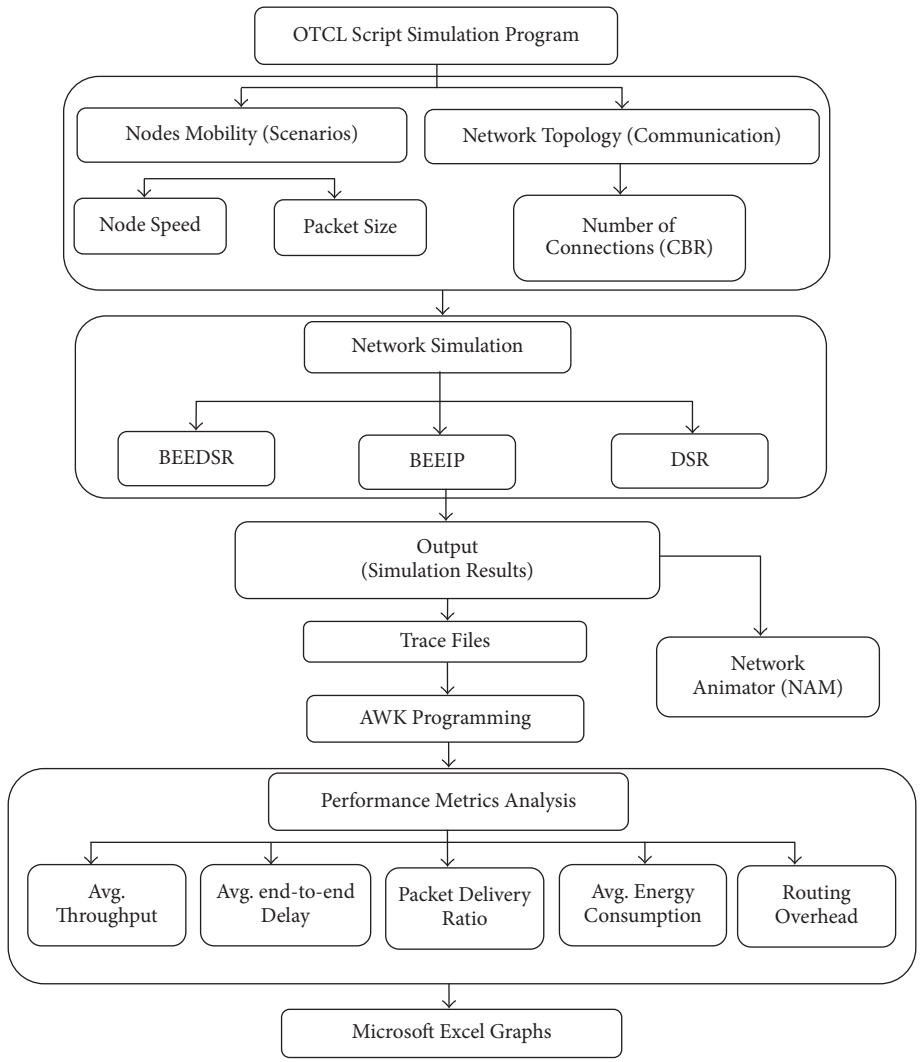

Figure 9: Flowchart of BEEDSR, BeeIP, and DSR execution in NS2.

Table 1: Simulation parameters.

Parameter	Value	Unit
Number of runs	10	—
Number of nodes	50	Node
Queue size	100	Packet
Simulation area	1670 × 970	m^2
Routing protocols	DSR, BeeIP, and BEEDSR	—
Mobility model	Random way point	—
Packet size	256	Bytes
Transmission range	250	M
Type of traffic	CBR	—
Initial energy	100	Joules
Idle power consumption	0.05	mW
Transmission power consumption	1.35	mW
Receive power consumption	1.7	mW
Sleep power	0.001	mW
Simulation time	50	Sec

$$\text{Average End-to-End Delay} = \frac{\sum_{i-1}^{n} (R.T_i - S.T_i)}{n}, \quad (2)$$

where $R.T_i$ is the total packet received, $S.T_i$ is the total packet sent, and n is the number of data packets.

5.2. Average Throughput. The average throughput metric represents the average of the successful data packets received to the total simulation time duration. The average throughput is measured in kilobits per second (kbps) and measures the effectiveness and efficiency of the routing protocol in receiving data packets by the destinations. The following formula is used to calculate the average throughput:

$$\text{Average Throughput} = \frac{\sum \text{Packets received by destinations}}{\text{stop time} - \text{start time}} * \frac{8}{1000}. \quad (3)$$

5.3. Routing Overhead Ratio. Routing overhead ratio metric represents the ratio of the total number of routing packets sent to the total number of routing and data packets sent. This metric provides an idea regarding the extra bandwidth consumed by the overhead to deliver data traffic. The routing overhead is computed using the following formula:

$$\text{Routing overhead Ratio} = \frac{\text{No of routing packets}}{\text{No of routing packets} + \text{No of data packets sent}} * 100. \quad (4)$$

5.4. Packet Delivery Ratio. The packet delivery ratio metric shows the total number of received data packets by destinations divided by the total number of data packets sent by the sources. This metric presents how a protocol successfully delivers packets from the source to the destination. A high packet delivery ratio indicates good results, which represent the wholeness and correctness of the routing protocol. The packet delivery ratio is computed using the following formula:

$$\text{Packet delivery Ratio} = \frac{\sum \text{packets received by destinations}}{\sum \text{packets sent by sources}} * 100. \quad (5)$$

5.5. Average Energy Consumption. This metric is measured as the ratio of total energy consumed by each node in the network divided by its initial energy. The initial and final energy left in the node are measured at the end of the simulation run. The average energy consumption is computed as follows:

$$\text{Average energy consumption} = \frac{\sum \text{energy consumed}}{\sum \text{initial energy}} * 100. \quad (6)$$

6. Simulation Results

6.1. Effects of the Node Speed. Figure 10(a) presents the variations of average energy consumption for BEEDSR, BeeIP, and DSR. These routing protocols have different energy consumption with increasing node speeds in the network. When the node speed varies from 5 to 20 m/s, BEEDSR energy consumption increased from 2.353037 to 2.428212, BeeIP from 2.359598 to 2.441357, and DSR from 2.360493 to 2.45088. However, BEEDSR has less average energy consumption than both the BeeIP and DSR protocols. The BEEDSR routing protocol reduced energy consumption by 0.47% and 0.79% compared with the BeeIP and DSR protocols. BEEDSR has a standard deviation of 0.033 for energy consumption, while BeeIP and DSR have standard deviations of 0.036 and 0.040, respectively.

The reason is that the BEEDSR protocol uses the ABC algorithm to redirect the data from the source to the destination by enhancing the existing DSR protocol. This algorithm selects the path that is less distant among each node as well as the shortest path. The shortest and best available path is then determined. Hence, the prior energy consumption on the route discovery on every node is reduced because the algorithm keeps track of the route from its memory.

Figure 10(b) shows the variations of the average end-to-end delay for BEEDSR, BeeIP, and DSR. When the node speed increases, the average end-to-end delay increases. BEEDSR increased from 13.922 to 27.807 ms, BeeIP from 37.109 to 37.983 ms, and DSR from 31.336 to 32.201 ms. However, the results clearly show that BEEDSR has better performance in terms of end-to-end delay than other routing protocols. BEEDSR achieved 67.46% and 41.79% delay reduction more than BeeIP and DSR protocols. The average end-to-end delay standard deviations for BEEDSR is 5.893, while BeeIP and DSR have standard deviations of 0.505 and 0.835, respectively. BEEDSR algorithm keeps the record of the shortest path from the source to the destination; thus, the time spent on discovering the route at each intermediate node for every transmission is no longer necessary. BEEDSR reduces the time consumed for transmission, which also reduces the packet drop ratio. The retransmission overhead is reduced, thereby resulting in faster data transmission in BEEDSR.

Figure 10(c) shows the average throughput for BEEDSR, BeeIP, and DSR. These routing protocols have different results in average throughputs with increasing number of node speeds in the network. When the node speeds increase from 5 to 20 m/s, BEEDSR decreased from 48.61704 to 46.16 kbps, BeeIP from 44.1916 to 43.8177 kbps, and DSR from 43.2925 to 40.92944 kbps. These results clearly indicate that BEEDSR is better in terms of the average throughput than other routing protocols. The reason is that the node speed varies and the packet drop ratio increases in both DSR and BeeIP. BEEDSR achieved a decrement of 7.05% and 12.30% more than BeeIP and DSR. BEEDSR has a standard deviation of 1.009 for average throughput, while BeeIP and DSR have standard deviations of 0.196 and 1.173, respectively. Once the packets reach their destination without any delay and without any drop, the channel bandwidth is utilized properly, thereby leading to high throughput.

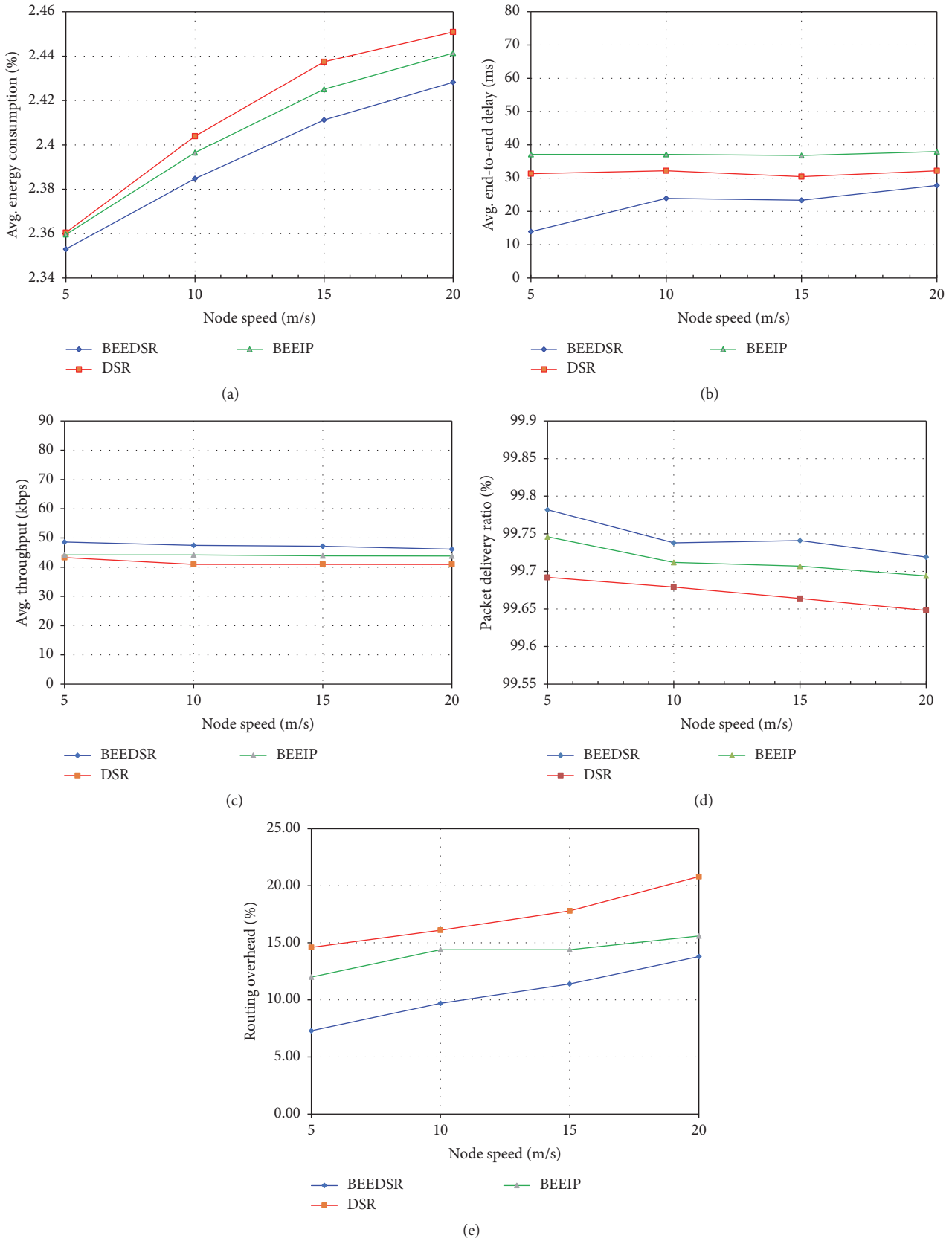

Figure 10: BEEDSR, DSR, and BeeIP performances versus node speed. (a) Average energy consumption, (b) average end-to-end delay, (c) average throughput, (d) packet delivery ratio, and (e) routing overhead ratio.

Figure 10(d) presents the packet delivery ratio for BEEDSR, BeeIP, and DSR. These routing protocols have different packet delivery ratios with increasing number of node speeds in the network. When the node speeds increase from 5 to 20 m/s, BEEDSR decreased from 99.782% to 99.719%, BeeIP from 99.749% to 99.694%, and DSR from 99.692 to 99.648%. However, BEEDSR has better result than BeeIP and DSR. BEEDSR achieved 0.030% and 0.074% decrement more than BeeIP and DSR. The packet delivery ratio standard deviations for BEEDSR is 0.027, while BeeIP and DSR have standard deviations of 0.022 and 0.019, respectively. The reason behind this is that the BEEDSR keeps track of its path and becomes aware of it once discovered. Data transfer also merely follows the path discovered by the protocol; thus, the packet delivery ratio remains high because most of the packets are delivered successfully. Moreover, minimal retransmission and drop ratio improve the high throughput.

Figure 10(e) shows the routing overhead for BEEDSR, BeeIP, and DSR. With the increasing speed of a node, all three protocols showed an increment routing overhead in the network. When the node speeds increase from 5 to 20 m/s, BEEDSR increased from 7.3% to 13.8%, BeeIP from 12% to 15.6%, and DSR from 14.6% to 20.8%. The reason for these changes is the high delivery rate of the packets. Specifically, the overhead of the retransmission and missing segments are both reduced because the packet drop ratio is considerably insignificant. Once the route is discovered, the same path is also used for further data transfer, thereby resulting in BEEDSR obtaining a minimal routing overhead compared with both BeeIP and DSR. BEEDSR achieved 20.99% and 53.46% overhead reduction more than BeeIP and DSR. BEEDSR has a standard deviation of 2.743 for routing overhead, while BeeIP and DSR have standard deviations of 1.501 and 2.660, respectively. The details of simulation results for BEEDSR, DSR, and BeeIP with respect to effect of node speed are shown in Table 2.

Table 2: Simulation results of node speed effect.

Speed (m/s)	BEEDSR	BEEIP	DSR
Average energy consumption (%)			
5	2.353037	2.359598	2.360493
10	2.384789	2.396493	2.403905
15	2.411249	2.425047	2.437463
20	2.428212	2.441357	2.45088
Average end-to-end delay (ms)			
5	13.922	37.109	31.336
10	23.902	37.118	32.192
15	23.357	36.81	30.448
20	27.807	37.983	32.201
Average throughput (kbps)			
5	48.61704	44.1916	43.2925
10	47.46709	44.1828	40.96004
15	47.1921	43.8844	40.95078
20	46.16	43.8177	40.92944
Packet delivery (%)			
5	99.782	99.746	99.692
10	99.738	99.712	99.679
15	99.741	99.707	99.664
20	99.719	99.694	99.648
Routing overhead (%)			
5	7.3	12.000	14.6
10	9.7	14.4	16.1
15	11.4	14.4	17.8
20	13.8	15.6	20.8

6.2. Effects of the Packet Size. Figure 11(a) shows the variations of the average energy consumption for BEEDSR, BeeIP, and DSR as the function of packet size. As the packet size increases for all protocols, the average energy consumption increases. Four different packet sizes were examined to measure the quality of the path selected using BEEDSR. When the packet sizes increase from 128 to 1,024 bytes, the BEEDSR energy consumption increased from 2.380943% to 2.615006%, BeeIP from 2.384978% to 2.689021%, and DSR from 2.417476% to 2.791907%. However, the results clearly show that BEEDSR has better performance in terms of average energy consumption than both BeeIP and DSR. BEEDSR achieved 1.52% and 4.74% energy consumption reduction more than BeeIP and DSR. BEEDSR has a standard deviation of 0.011 for average energy consumption, while BeeIP and DSR have standard deviations of 0.137 and 0.159, respectively. Once the path is discovered by the protocol, the same path is used to transmit all the packets. This path is the shortest and has a minimum hop count. Therefore, the intermediate nodes do not have to consume additional energy and waste battery power because route discovery is unnecessary.

Figure 11(b) shows the average end-to-end delay for BEEDSR, BeeIP, and DSR. These routing protocols have different end-to-end delays with the increasing number of packet sizes in the network. All three test protocols show an increase in average end-to-end delay as the packet size increases. When the packet sizes increase from 128 to 1024 bytes, BEEDSR increased from 27.7 to 60.151 ms, BeeIP from 31.942 to 70.853 ms, and DSR from 33.665 to 77.727 ms. However, BEEDSR has less delay than both BeeIP and DSR. BEEDSR achieved 21.33% and 31.93% delay reduction compared with BeeIP and DSR. The average end-to-end delay standard deviations for BEEDSR is 14.694, while BeeIP and DSR have standard deviations of 19.791 and 17.279, respectively. Most of the packets are delivered on a well-suited path; thus, they help reduce the end-to-end delay in BEEDSR. BEEDSR keeps maintaining up-to-date routing information. In case of link failure caused by the changeable location of a node upon acceleration, an alternative path is made available and can be immediately used, thereby decreasing the average end-to-end delay.

Figure 11(c) shows the variations in the average throughput for BEEDSR, BeeIP, and DSR. With the increment in packet size, all three test protocols show an increase in average throughput. When the packet sizes increase from 128 to

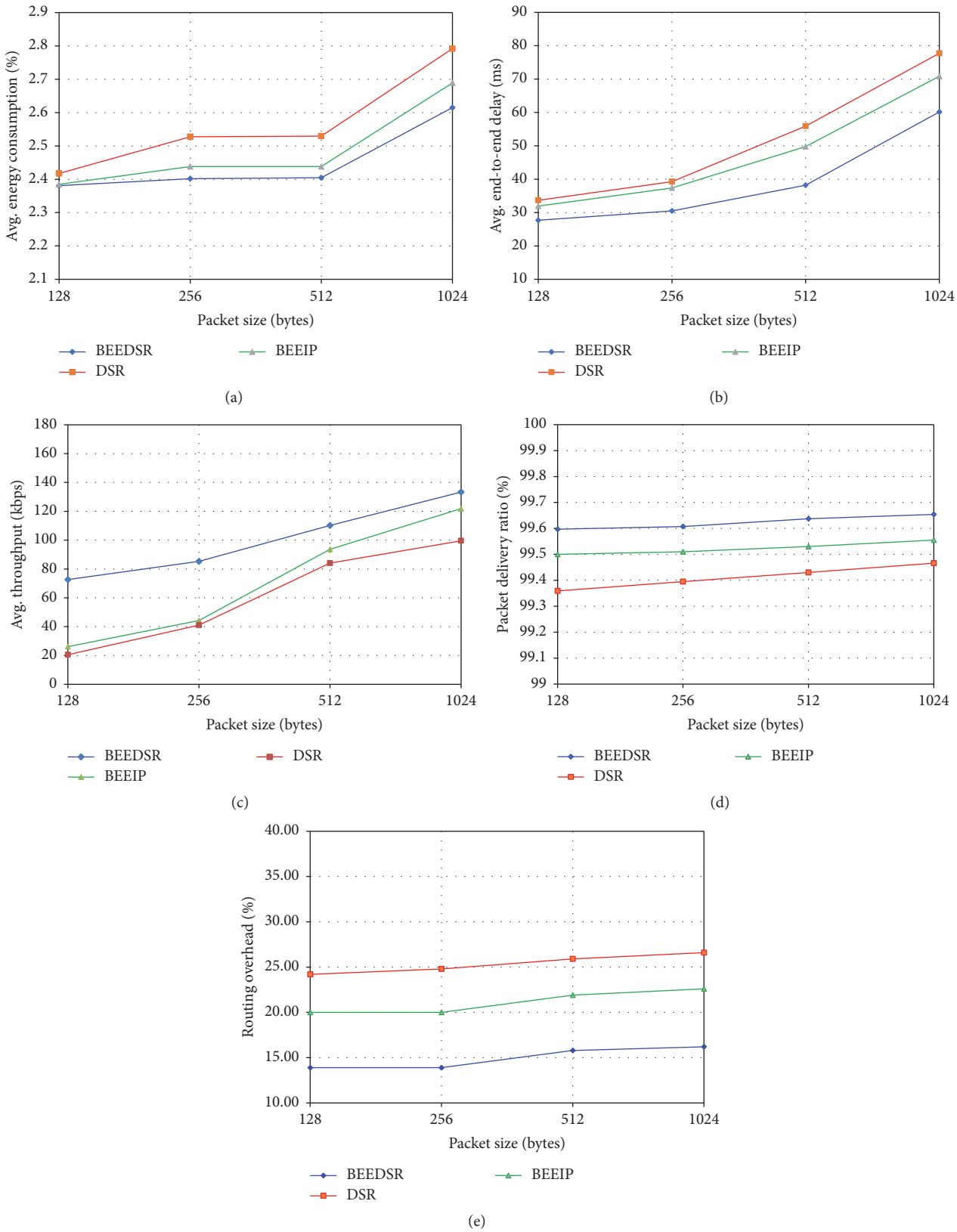

Figure 11: BEEDSR, DSR, and BeeIP performances versus packet size. (a) Average energy consumption, (b) average end-to-end delay, (c) average throughput, (d) packet delivery ratio, and (e) routing overhead ratio.

1,024 bytes, BEEDSR throughput increased from 72.6411 to 133.3468 kbps, BeeIP from 26.08562 to 121.8802 kbps, and DSR from 20.52148 kbps to 99.554379 kbps. These results clearly indicate that BEEDSR has better performance in terms of the average throughput compared with both BeeIP and DSR. BEEDSR achieved 28.80% and 38.93% throughput increment more than BeeIP and DSR. BEEDSR has a standard deviation of 26.962 for average throughput, while BeeIP and DSR have standard deviations of 44.106 and 36.750, respectively. The reason is that the reactive routing mechanism is established with BEEDSR; thus, more than one RREP for each RREQ and BEEDSR keep track of the path between the source and the destination. An alternative path is selected in case one path is broken. Hence, the average throughput of BEEDSR increases considerably as the number of packets getting dropped is reduced.

Figure 11(d) shows the packet delivery ratios for BEEDSR, BeeIP, and DSR. These routing protocols have different packet delivery ratios with the increasing number of packet sizes in the network. When the packet sizes increase from 128 to 1,024 bytes, BEEDSR packet delivery ratio increased from 99.579% to 99.654%, BeeIP from 99.500% to 99.555%, and DSR from 99.359% to 99.466%. However, BEEDSR has better results in terms of packet delivery ratio than both BeeIP and DSR. The BEEDSR achieved 0.10% and 0.21% packet delivery improvement more than the BeeIP protocol and DSR. The packet delivery ratios' standard deviations for BEEDSR are 0.026, while BeeIP and DSR have standard deviations of 0.024 and 0.046, respectively.

Based on ABC colony, the BEEDSR keeps track of and updates its path once the shortest and best path is discovered; the rest of the data packets transfer follows the same path between the source and destination with no overhead. Therefore, the number of packets getting dropped is reduced, and the packet delivery ratio is increased. In DSR and BeeIP, when a source node aims to send a packet to a destination, it searches for its route cache to ensure if it already contains a route to the destination.

Figure 11(e) shows the routing overhead for the three kinds of routing protocols BEEDSR, BeeIP, and DSR. Notably, in the increasing packet size, all three test protocols show an increase in the routing overhead. BEEDSR increased from 13.900% to 16.200%, BeeIP from 20.000% to 22.600%, and DSR from 24.200% to 26.600%. BEEDSR achieved 20.71% and 54.62% overhead reduction more than BeeIP protocol and DSR, respectively. BEEDSR has a standard deviation of 1.223 for routing overhead, while BeeIP and DSR have standard deviations of 1.330 and 1.078, respectively. The results clearly indicate that BEEDSR has better performance in terms of the routing overhead than both BeeIP and DSR. In BEEDSR, the retransmission of certain packets is no longer necessary because the number of dropped packets is reduced; thus, the protocol overhead is reduced.

The details of simulation results for BEEDSR, DSR, and BeeIP with respect to packet size are shown in Table 3.

Table 3: Simulation results of packet size effect.

Packet size (bytes)	BEEDSR	BEEIP	DSR
Average energy consumption (%)			
128	2.380943	2.384978	2.417476
256	2.401592	2.43839	2.527688
512	2.404601	2.438344	2.52943
1024	2.615006	2.689021	2.791907
Average end-to-end delay (ms)			
128	27.7	31.942	33.665
256	30.511	37.348	39.252
512	38.175	49.783	55.885
1024	60.151	70.853	77.727
Average throughput (kbps)			
128	72.6411	26.08562	20.52148
256	85.26032	44.21113	41.02791
512	110.1801	93.65283	84.02914
1024	133.3468	121.8802	99.554379
Packet delivery (%)			
128	99.597	99.500	99.359
256	99.607	99.510	99.395
512	99.637	99.530	99.430
1024	99.654	99.555	99.466
Routing overhead (%)			
128	13.900	20.000	24.200
256	13.900	20.000	24.800
512	15.800	21.900	25.900
1024	16.200	22.600	26.600

7. Conclusion and Future Work

The successful use of ABC algorithm in many MANET applications motivated this research to adapt the same algorithm to improve the DSR routing protocol. This study provides the optimization of the current DSR routing over MANET. The proposed BEEDSR routing protocol is inspired from the natural bee food hunting behavior to overcome the energy problems caused from overload packet from source to destination MANET nodes. The BEEDSR routing technique focuses on determining the optimal routing path. The advantage of BEEDSR is its simplicity; this routing protocol can be easily integrated into existing ad hoc routing algorithms without affecting other communication protocol layers. The simulation results for BEEDSR demonstrate noticeable improvements under wide network parameter settings, such as node speed and packet size.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported in part by Effat University, Jeddah, Saudi Arabia, under Internal Research Grant Scheme, Grant no. UC#7/02.MAR/2016/10.2-20a.

References

[1] G. Varghese, "Life in the fast lane-viewed from the confluence lens," *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 19–25, 2015.

[2] A. K. Pandey and H. Fujinoki, "Study of MANET routing protocols by GloMoSim simulator," *International Journal of Network Management*, vol. 15, no. 6, pp. 393–410, 2005.

[3] M. R. Zasad and J. Uddin, *Study and performance comparison of MANET routing protocols:TORA, LDR and ZRP, Master Thesis [Master, thesis]*, Blekinge Institute of Technology, Sweden, 2010.

[4] X. Hong, K. Xu, and M. Gerla, "Scalable routing protocols for mobile ad hoc networks," *IEEE Network*, vol. 16, no. 4, pp. 11–21, 2002.

[5] X. Zhao, W. N. N. Hung, Y. Yang, and X. Song, "Optimizing communication in mobile ad hoc network clustering," *Computers in Industry*, vol. 64, no. 7, pp. 849–853, 2013.

[6] S. Choudhary and S. Jain, "A survey of energy-efficient fair routing in MANET," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 1, pp. 416–421, 2015.

[7] J.-Z. Sun, "Mobile ad hoc networking: an essential technology for pervasive computing," in *Proceedings of the International Conferences on Info-Tech and Info-Net, ICII 2001*, pp. 316–321, November 2001.

[8] B. Ishibashi and R. Boutaba, "Topology and mobility considerations in mobile ad hoc networks," *Ad Hoc Networks*, vol. 3, no. 6, pp. 762–776, 2005.

[9] A. Zadin and T. Fevens, "Maintaining path stability with node failure in mobile ad hoc networks," *Procedia Computer Science*, vol. 19, pp. 1068–1073, 2013.

[10] D. J. Rao, K. Sreenu, and P. Kalpana, "A study on dynamic source routing protocol for wireless ad hoc networks," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, pp. 2319–5940, 2012.

[11] S. M. Onyemelukwe, *Evaluation of on-demand routing in mobile ad hoc networks and proposal for a secure routing protocol [M.sc. thesis]*, Electrical and Computer Engineering, University of Windsor, Ontario, Canada, 2013.

[12] D. Johnson, Y. Hu, and D. Maltz, "The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4," RFC 4728, IETF, 2007.

[13] Y. Ravikumar and S. K. Chittamuru, *A case study on MANET routing protocols performance over TCP and HTTP [Master, thesis]*, School of Engineering, Blekinge Institute of Technology, Master Thesis Electrical Engineering, MSE-2010-6434, 2010.

[14] D. Sivakumar, B. Suseela, and R. Varadharajan, "A survey of routing algorithms for MANET," in *Proceedings of the 1st International Conference on Advances in Engineering, Science and Management, ICAESM-2012*, pp. 625–640, ind, March 2012.

[15] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, 2014.

[16] A. S. Bhagade and P. V. Puranik, "Artificial bee colony (ABC) algorithm for vehicle routing optimization problem," *International Journal of Soft Computing and Engineering*, vol. 2, pp. 329–333, 2012.

[17] H. Duan and P. Li, *Bio-inspired computation in unmanned aerial vehicles*, Springer, Heidelberg, 2014.

[18] A. Giagkos and M. S. Wilson, "BeeIP - A Swarm Intelligence based routing for wireless ad hoc networks," *Information Sciences*, vol. 265, pp. 23–35, 2014.

[19] H. Sridhar, M. Siddappa, and G. B. Prakash, "Power aware routing protocol for MANET's using swarm intelligence," *nternational Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, pp. 2960–2965, 2013.

[20] K. Santhiya and N. Arumugam, "Energy Aware Reliable Routing Protocol (EARRP) for Mobile Ad Hoc Networks Using Bee Foraging Behavior and Ant Colony Optimization," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, pp. 1694-0814, 2012.

[21] B. C. Mohan and R. Baskaran, "Energy aware and energy efficient routing protocol for adhoc network using restructured artificial bee colony system," *Communications in Computer and Information Science*, vol. 169, pp. 473–484, 2011.

[22] M. I. Fahmy, L. Nassef, and A. H. Hefny, "On the performance of the predicted energy efficient bee-inspired routing (PEEBR)," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 5, no. 4, pp. 65–70, 2014.