Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 6421717, 19 pages
https://doi.org/10.1155/2018/6421717

Research Article

Congestion Control and Prediction Schemes Using Fuzzy Logic System with Adaptive Membership Function in Wireless Sensor Networks

Phet Aimtongkham,[1] **Tri Gia Nguyen**,[2] **and Chakchai So-In**[1]

[1]*Applied Network Technology (ANT) Laboratory, Department of Computer Science, Faculty of Science, Khon Kaen University, Khon Kaen, Thailand*
[2]*Department of Information Technology, Duy Tan University, Vietnam*

Correspondence should be addressed to Chakchai So-In; so-in@ieee.org

Received 27 March 2018; Revised 21 June 2018; Accepted 4 July 2018; Published 1 August 2018

Academic Editor: Al-Sakib K. Pathan

Network congestion is a key challenge in resource-constrained networks, particularly those with limited bandwidth to accommodate high-volume data transmission, which causes unfavorable quality of service, including effects such as packet loss and low throughput. This challenge is crucial in wireless sensor networks (WSNs) with restrictions and constraints, including limited computing power, memory, and transmission due to self-contained batteries, which limit sensor node lifetime. Determining a path to avoid congested routes can prolong the network. Thus, we present a path determination architecture for WSNs that takes congestion into account. The architecture is divided into 3 stages, excluding the final criteria for path determination: (1) initial path construction in a top-down hierarchical structure, (2) path derivation with energy-aware assisted routing, and (3) congestion prediction using exponential smoothing. With several factors, such as hop count, remaining energy, buffer occupancy, and forwarding rate, we apply fuzzy logic systems to determine proper weights among those factors in addition to optimizing the weight over the membership functions using a bat algorithm. The simulation results indicate the superior performance of the proposed method in terms of high throughput, low packet loss, balancing the overall energy consumption, and prolonging the network lifetime compared to state-of-the-art protocols.

1. Introduction

Recently, according to Gardner [1], over 8 billion devices have connected and interacted to form the Internet of Things (IoT) network, most of which perform sensing and monitoring tasks [2] and report their data over a cloud-based platform for further analysis using big data technology [3]. One key technology involved in this trend is the use of wireless sensor networks (WSNs) [4].

In WSNs, many sensing devices or sensor nodes (SNs) are connected, typically via wireless transmissions (e.g., radio frequency (RF)) to overcome the limitation of wire-based infrastructure. These SNs can form an ad hoc network (multihop transmissions via relays or intermediate SNs) that is self-computing, self-contained, and self-organized and that can interact with nodes that are outside the network toward the gateway (sink node or base station (BS)) [5].

Although SNs involve multifunctional devices, including sensing, computing, storage, and transmission units, the key limitation is in their energy consumption. The self-power supply of the battery determines the SN lifetime. In addition to the several advantages of WSNs, there are multiple challenges, such as routing, localization, quality of service (QoS), and security [6, 7]. In particular, the traffic over the network is massive in large-scale WSNs, causing problems with data transmission in determining the optimal path to the sink node as well as heterogeneous traffic in various topologies, especially nonuniform traffic once randomly deployed to places such as forests and disaster areas [8, 9], thereby shortens the overall network lifetime, and results in

congestion control issues in WSNs [10, 11] for applications such as thermal rating monitoring of conductors in power systems, life sensor monitoring in health-care applications, and earthquake warning systems [12, 13].

Traditionally, congestion is one of the main issues affecting network outperformance, especially with high density nodes from wired [14] to wireless [15] and particularly WSN. Different from others, WSN has specific characteristics, such as very low energy via battery operation and low bandwidth, which are of great concern for congestion [16]. In general, two main approaches are used to mitigate congestion in WSNs: traffic control (TC) and resource control (RC) [17, 18]. The former is used to adjust the transmission rate, e.g., additive increase multiplicative decrease (AIMD). This technique regularly measures the current bandwidth and adjusts the rate based on the presence of transmitted SNs with consideration of fairness. The key advantage of TC is the sudden effect of congestion alleviation with the decrease of sending rates; however, the main limitations are probable low throughput and packet loss during the initial stage of congestion.

Generally, there are three types of congestion [19]: at the source node (caused by the simultaneous transmission of data by SNs that are close to one another), at an intermediate node (caused by data aggregation and the parent node section strategy, in particular, with buffer overflow), and at the sink node (caused by too many children sending traffic to the only sink node at the same time). These types of congestion can cause packet loss, delay, and low throughput.

The latter approach, RC, mainly refers to an approach that is not limited to decreasing the sending rate; however, in wireless communications, more than one path is usually directed to the destination (BS). Alternate path selection (in which a detour to the destination is constructed with available relays or SNs) is one type of RC. In this method, the key advantage is to balance the resources in the network (i.e., the available SNs), which results in higher throughput. This approach is our focus in the present research.

With an RC-based approach, several factors can affect the network performance, particularly the congestion mitigation, such as buffer occupancy, transmission rate, distance between the sender and the receiver, and most importantly, energy. There is no linear relationship among these factors. Therefore, despite the advantages of RC-based approaches, packet loss and low throughput are still probable due to unsuitable path selection, given those factors.

Thus, we focus on enhancing the RC by considering a probable congestion prediction; in particular, we seek the best alternate path before congestion using a fuzzy logic system (FLS) for adaptive path determination in consideration of buffer occupancy, forwarding rate, hop count, and remaining energy, called the Optimized Fuzzy Logic-based Congestion Control Scheme with Exponential Smoothing Prediction in WSNs (OFES). Our contributions are as follows:

(i) Propose an architecture for congestion mitigation in WSNs.

(ii) Adopt exponential smoothing to predict the buffer occupancy.

(iii) Apply FLS to determine appropriate weights for path determination in consideration of hop count, remaining energy, buffer occupancy, and forwarding rate.

(iv) Optimize FLS with a bat algorithm (BAT) to tune the membership function.

The remainder of the article is organized as follows: In Section 2, we review related work on congestion controls and focus on resource-control-based approaches and fuzzy integration. Section 3 provides the assumptions and definitions of our wireless sensor network models. In Section 4, our proposed approach, which is based on a fuzzy-based congestion control and prediction scheme, is presented. Section 5 presents and discusses the performance evaluation through simulation results. Finally, the conclusions and the future research direction are discussed in Section 6.

2. Related Work

In this section, we focus on RC-based congestion controls in WSNs. We also discuss work on FLS-based congestion controls and predictions.

The pioneering approach to RC is Topology-Aware Resource Adaptation for Congestion Avoidance (TARA), which was proposed by J. Kang et al. [20]. The authors proposed a topological control that applies the concept of distributed and merged pathways. Once congestion is detected, a particular node will split the path (into multipaths) to detour the direct transmission to the congested SN. Then, the multipaths are combined at the merged SN. However, the alternate path can also encounter congestion. More paths can lead to high energy consumption by the network.

Different approaches were considered by only focusing on the best-possible path, such as the Hierarchical Tree Alternative Path (HTAP), proposed by C. Sergiou et al. [21]. Starting from the sink node, a hop-by-hop-based approach is applied, which uses the buffer occupancy information as a threshold. This approach forms a hierarchical tree structure from the source node. However, it may cause a delay since the SN must determine the possible path in each hop-by-hop transmission according to the congestion.

To overcome the limitations of HTAP, the same set of authors proposed a modified approach, the Dynamic Alternative Path Selection Scheme (DAiPAS) [22], by initially constructing the topology to avoid congestion in the hop-by-hop-based scheme. DAiPAS also works with soft and hard states. A soft state tends to have only 1 communication link between each SN and its parent, which is formed by announcing the presence of a transmission. Therefore, the other SNs to be transmitted (if any) should seek other paths before congestion occurs. In contrast, in a hard state, the buffer occupancy will be used to determine the acceptance of an SN by its parent transmission if there is no option.

W. Chen et al. [23] developed the Congestion Control and Energy-balanced scheme based on the Hierarchy (CCEbH), which has an awareness of energy balance. CCEbH adopts the initial topology construction from DAiPAS. However, there is a threshold (quota) for each SN transmission to its parent, i.e., a time interval for each transmission (based on the buffer occupancy and the forwarding rate). There is also an

energy balance method in two dimensions, i.e., vertical and horizontal. The SN will typically ask its vertical and horizontal neighbors to determine the paths to the sink node. Here, time synchronization may become an issue in high traffic and dense deployment.

In addition to its use in RC-based methods, FLS is typically used for TC-based approaches. For example, S. Jaiswal and A. Yadav proposed Fuzzy-Based Adaptive Congestion Control (FBACC) [24]. Some of the key parameters of FBACC are related to probable congestions. For example, buffer occupancy, forwarding rate, and the number of connections of a (to be) congested SN are used as the inputs for FLS to determine the appropriate sending rate. However, here, there is no consideration of the energy of SNs as a key factor for prolonging the network lifetime of the WSN.

Y. Zhang et al. [25] considered the energy of SNs, which includes the energies of a particular SN and its neighbors and the path information, in the level-based path determination as the inputs of FLS for selecting the cluster head (CH); this method is named the Energy-Efficient Distributed Clustering algorithm based on fuzzy approach with nonuniform distribution (EEDCF). The CH transmits to the next CH toward the sink node. The main limitation is that, in the case of sparse deployment, EEDCF tends to underperform.

Based on the pioneering work on the TC-based scheme Congestion Detection and Avoidance (CODA) [26], while applying backward congestion notification to the source node to adjust the sending rate to mitigate the effects of congestion, research has also been conducted on congestion prediction to avoid congestion. For example, M. Zawodniok et al. [27] proposed Dynamic Predictive Congestion Control (DPCC). DPCC utilizes queuing, including the embedded channel estimator algorithm, as the metric for adjusting the sending rate; however, this approach incurs high overhead. A. U. Rajan et al. proposed the Rate Regulation Split Protocol (RRG) [28] with the concept of congestion prediction, which uses a probabilistic method on data traffic and buffer occupancy to determine a congestion probability.

3. Models and Assumptions

In this section, we describe networking and energy models, including all required messages, as follows.

3.1. Network Models. In this research, similar to other recent work on congestion control in WSN [20–28], there is only one base station or sink node, which is typically placed at the end of the network (to be connected to the Internet). There is a power source at this node. At a specific time, sensor nodes that are activated by certain events (such as sensing and becoming ready for data transmission), also known as source nodes, will send data toward the sink node via the next or neighboring nodes (hop-based), nonuniform deployments. We make the following network assumptions:

(i) Sensor nodes (SNs): in addition to the sink node, SNs are randomly deployed in a uniform manner within a network square area, each of which has the same initial energy.

(ii) After the SNs are deployed, there is no mobility involved and no explicit location information.

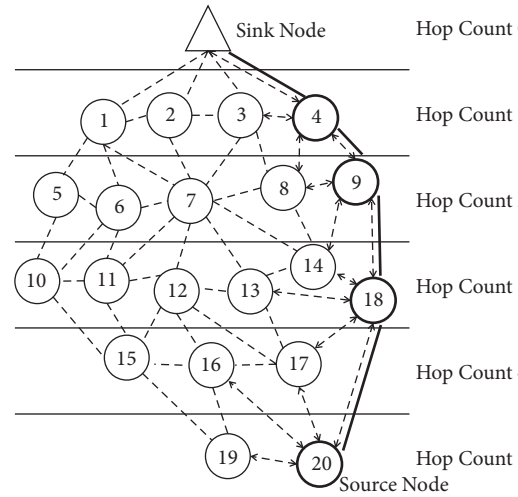


Figure 1: Example: network model.

(iii) Each SN has the same communication range.

(iv) All SNs within the network have been given the time-synchronization protocol for communication purposes [20–28].

(v) Any SN can leave the network if and only if the energy is depleted (it is no longer in operation).

Figure 1 shows an example of our network model, which consists of a sink, a source, and other SNs. The dashed line defines the communication range and its connectivity from each SN to its neighboring node (NN). The arrow head is determined by the broadcast message. The network will be generally divided into layers based on the number of hop counts. Once the path has been defined (as the solid line), the source node will send the data to the sink node accordingly (from the 20th SN, go over 18, 9, and 4, and finally go to the sink node).

3.2. Energy Models. To communicate within WSNs, we follow a standard IEEE 802.15.4 Low-rate Wireless Personal Area Network (LR-WPAN) protocol, in which the energy depends on the number of packets that are sent and received by a specific SN. Equation (1) shows the energy model for WSNs [29], where the transmission energy E_{Tx} (Joules/bit) of the ith bit over a distance d in the free space propagation model is expressed as follows:

$$E_{Tx}(i,d) = i \cdot E_{elec} + i \cdot E_{tran} \cdot d^2 \quad (1)$$

where E_{elec} denotes the data processing energy and E_{tran} is the energy for the transmission of 1 bit over a distance d.

Similar to the transmission, (2) expresses the receiving energy E_{Rx} (Joules/bit) of i bits, where E_{Reci} is the energy for receiving 1 bit.

$$E_{Rx}(i) = i \cdot E_{Reci} \quad (2)$$

Note that, based on the specification of MicaZ [30] with 3 volts (2 AA batteries), $E_{elec} = 0.96\ \mu$J/bit, $E_{tran} = 0.144\ \mu$J/bit, and $E_{reci} = 0.96\ \mu$J/bit.

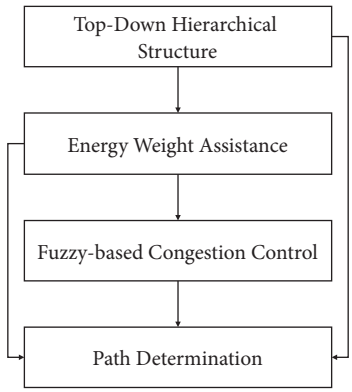


Figure 2: Overview of OFES.

3.3. System Messages. There are eight types of messages in both broadcast and unicast for our OFES architecture:

(i) *Route-Request-Msg(Node_ID, Hop Count)*. This broadcast message is used to make a route query to neighbors, and it includes the node identifier and the hop count from the sink node.

(ii) *Route-Reply-Msg(Node_ID, Hop Count)*. Once *Route-Request-Msg* has been received, the neighbors reply with this broadcast message, which includes their identities and hop counts.

(iii) *Route-End-Msg(Node_ID, Hop Count)*. When no *Route-Reply-Msg* is sent back within a specific timeout, a specific node will send this broadcast message to indicate the end of the hop tree (to determine a maximum hop count).

(iv) *Route-Update-Msg(Node_ID)*. If a node does not have enough energy to perform an operation (within a threshold), that node will send this broadcast message to its neighbors, which includes its identity.

(v) *Request-Stage-Msg(Node_ID)*. This message is broadcasted to an SN's neighbors for weight inquiry.

(vi) *Reply-Stage-Msg(Node_ID, EW, AW)*. After *Request-Stage-Msg* has been received, each SN will send this unicast message back, which includes the identity and two additional (locally computed) weights: the Energy-Assisted Routing Weight (EW) and the Availability Weight (AW).

(vii) *Data-Sent-Msg(Node_ID, Data)*. This unicast message is used for data transmission after the final path has been determined.

(viii) *Data-Ack-Msg(Node_ID)*. This unicast message is used for data transmission confirmation.

4. Optimized Fuzzy Logic-Based Congestion Control Scheme with Exponential Smoothing Prediction in WSNs (OFES)

Figure 2 shows an overview of the OFES architecture. There are 4 main components:

Table 1: Example: neighbor table in Route Create.

Node ID	Hop Count

(i) **Top-Down Hierarchical Structure.** This component is first used to generate the layer-based structure with hop count as the key metric. We adopt the Ad hoc On-Demand Distance Vector (AODV) [31], which is used for noninfrastructure wireless networks, to generate a many-to-one structure from SNs to the sink node. Two main steps are applied as (1) Route Create and (2) Route Update phases.

(ii) **Energy-Assisted Routing Weight.** The second component is used for weight generation with FLS. Two fuzzy inputs are considered here: hop count and remaining energy.

(iii) **Congestion Control and Prediction.** This component is mainly used for congestion control to aid in congestion prediction. Here, we adopt exponential smoothing, with its optimization over FLS tuned by BAT, and the buffer occupancy and the forwarding rate as fuzzy inputs.

(iv) **Path Determination.** After weight derivation, this final component is used to determine a final path toward the sink node. Generally, with the performance versus energy consumption trade-off, OFES can select one of the three components to achieve the highest performance: applying only the first component, the first and the second components, or all three components.

4.1. Top-Down Hierarchical Structure. In this component, OFES starts to construct the topology for path determination by further applying the layering concept with hop count determination (one for each additional layer) from the sink node to the possible source nodes (all SNs) as the key metric. Two steps are carried out: the Route Create and Route Update phases.

4.1.1. Route Create Phase. We adopt AODV as a baseline for determining a layer-based approach, using hop count as a main metric. Similar to the tree-based concept, here, the sink node is considered a root of the tree and an additional layer after the root is traversed hop by hop (in a top-down manner) until reaching the source node.

First, the sink node (hop count = 0) broadcasts *Route-Request-Message* to the nearby nodes (neighbors). Once each neighboring node receives this message, it stores the information (Node ID and Hop Count) in the neighbor table (see Table 1) and increases the hop count by 1 to implicitly specify how far it is from the sink node.

After receiving and storing the information, this neighboring node sends the reply message *Route-Reply-Msg* to confirm its receipt. This node continues this process (*Route-Request-Message* and *Route-Reply-Msg*) until there are no additional neighboring nodes (no responses to *Route-Reply-Msg*). In this case, this node (the end node of a particular

Sink node broadcasts *Route-Request-Msg* to SN_i
while (*SN* is an intermediate node) do
SN_i receives *Route-Request-Msg* and creates [$NeighborTable_i$]
SN_i sends *Route-Reply-Msg* and broadcasts *Route-Request-Msg*
    if SN_i is last hierarchy node **then**
    SN_i *sendsRoute-End-Msg*
    Break
    end if
end while

Algorithm 1: Topology construction in Route Create.

Table 2: Example: Neighbor table of SN 4 in Route Create.

Node ID	Hop Count
(Sink Node)	0
3	1
8	2
9	2

Table 3: Example: neighbor table of SN 20 in Route Create.

Node ID	Hop Count
18	3
16	4
17	4
19	5

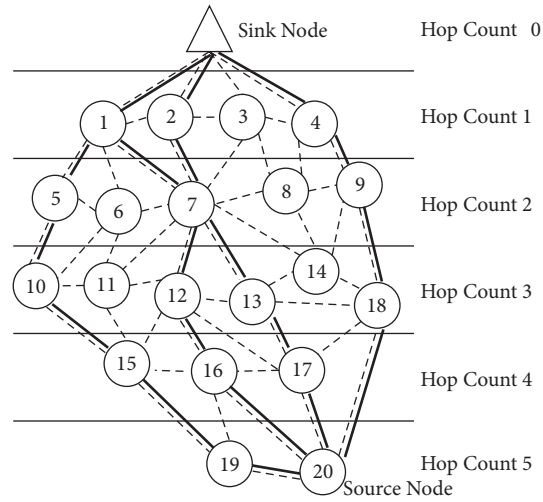


Figure 3: Example: possible path selection in Route Create.

path along the tree, Timeout 3,000 ms) sends *Route-End-Msg* back to the previous node (lower hop count). This process continues until the sink node is reached. Note that this message is used for maximum hop count determination of each path.

Note that, in case a particular node may receive more than one *Route-Request-Message*, the node will select (or update) the table based on the number of hop counts (lowest is preferred).

Table 2 shows a neighbor table of the 4th SN (see also Figure 1); here, there is only 1 neighbor (SN = 3rd) for the first broadcast and there are 2 neighbors (SNs = 8th and 9th) in the next broadcast. Note that, here, we assume the architecture is hop-by-hop communication. When the broadcasting pattern has been completed, the end node of the tree is node 20. The neighbor table is shown as Table 3.

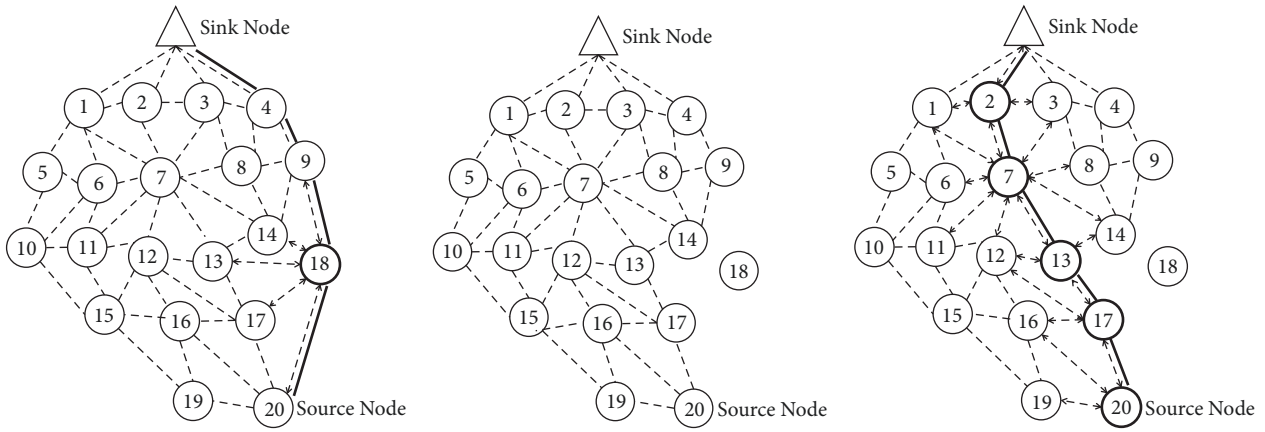
Figure 3 also shows an example of this case; here, there are six possible paths from the 20th SN to the sink node. Algorithm 1 also shows the steps of the Route Create phase. The final path determination will be discussed again in Section 4.4.

4.1.2. Route Update Phase. To avoid a disconnectivity, the Route Update phase is used to switch the route. Generally, the disconnectivity (disjoint path) is due to a lack of energy (when the battery has run out). Here, we specify an energy threshold for each SN. If the energy falls below the threshold (here, we set it to 5% of the initial energy [32]), the SN will send *Route-Update-Msg* to its neighbors to update the neighbor tables.

Figure 4 shows an example of this phase. Figure 4(a) shows the final determined path (the shortest path from 20 ⟶ 18 ⟶ 9 ⟶ 4 ⟶ sink node) from the possible paths (see Figure 3), which include (20 ⟶ 19 ⟶ 15 ⟶ 10 ⟶ 5 ⟶ 1 ⟶ sink node), (20 ⟶ 16 ⟶ 12 ⟶ 7 ⟶ 1 ⟶ sink node), (20 ⟶ 16 ⟶ 12 ⟶ 7 ⟶ 2 ⟶ sink node), (20 ⟶ 17 ⟶ 13 ⟶ 7 ⟶ 1 ⟶ sink node), (20 ⟶ 17 ⟶ 13 ⟶ 7 ⟶ 2 ⟶ sink node), and (20 ⟶ 18 ⟶ 9 ⟶ 4 ⟶ sink node). However, when the energy of the 18th SN is low, it will broadcast *Route-Update-Msg*, and its neighbors will update the table (shown in Figure 4(b)). Figure 4(c) shows the selected path: 20 ⟶ 17 ⟶ 13 ⟶ 7 ⟶ 2 ⟶ sink node (the next-shortest path). Table 4 shows the neighbor table in the absence of the 18th SN (see also Table 3 for the neighbor table in the presence of the 18th SN).

Figure 5 shows a detailed sequence diagram of both the Route Create and Route Update phases.

4.2. Energy-Assisted Routing Weight (EW). This component is used to determine routing criteria (as a weight or Energy-Assisted Routing Weight (EW)) between hop count and



(a) SN 18 sends *Route-Update-Msg* once the energy falls below the threshold

(b) Neighbors of SN 18 remove the connectivity from the table

(c) Final path determination from 20 ⟶ 17 ⟶ 13 ⟶ 7 ⟶ 2 ⟶ sink node

Figure 4: Example: Route Update at SN 18.

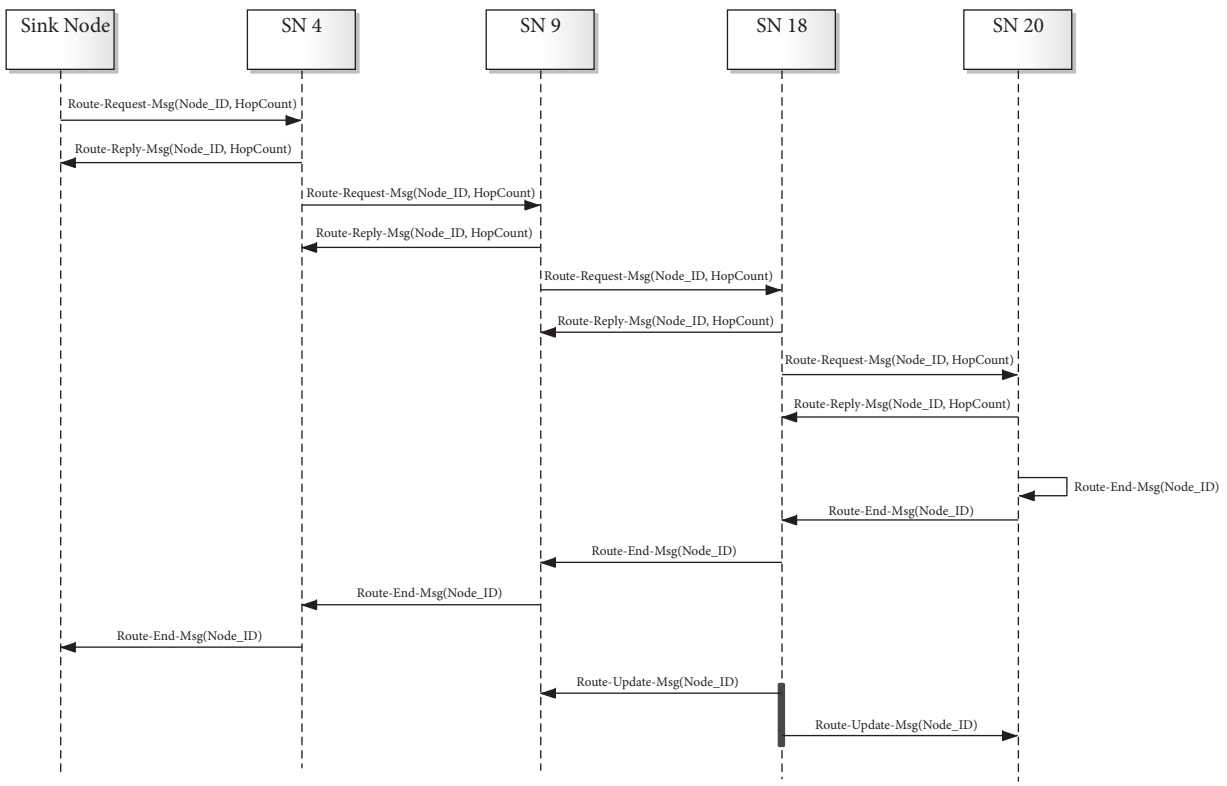


Figure 5: Sequence diagram of OFES in Route Create and Update phases.

Table 4: Example: Neighbor Table in SN 20 in the absence of the 18th SN.

Node ID	Hop Count
16	4
17	4
19	5

remaining energy. We applied FLS to determine a final weight based on the membership function.

(A) Fuzzy Logic System (FLS). FLS is a technique for logical reasoning with an approximation-based approach as opposed to an exact solution. FLS identifies solutions within the range between 0 and 1 (not exactly 0 or 1) due to the fuzziness. Based on our evaluation process, FL with Mamdani outperformed

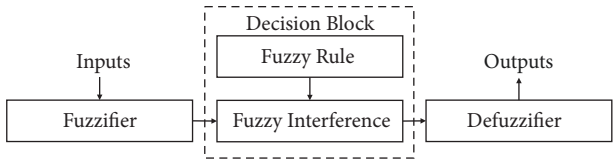


Figure 6: Block diagram: adaptive fuzzy logic system.

Table 5: Examples of 9 EW fuzzy rules.

IF (Hop Count)	AND (Remaining Energy)	THEN (Weight)
L	L	M
L	M	M
L	H	H
M	L	L
M	M	M
M	H	H
H	L	L
H	M	L
H	H	M

FL Sugeno; thus, we select FL with Mamdani, which has four main steps [33] (see Figure 6).

Fuzzifier is used to transform the input (i.e., hop count and remaining energy) to a numerical format, which is typically normalized to a range between 0 and 1 from the current over the maximum hops and similarly the remaining over the initial energy. Then, the input is mapped to determine the crossing points of each function (corresponding to fuzzy rules). Based on our intensive evaluation, we applied triangular (low, medium, and high for output weight) and trapezoidal (low, medium, and high for the two inputs) functions, which outperformed the others, such as Gaussian, generalized bell, and sigmoid. Figure 7 shows an example of a membership function of hop count and remaining energy, including the output weights.

Fuzzy rule is used to generate the input-to-output mapping based on the membership function (if/then rule). Table 5 shows a rule construction procedure with 9 rules, including a combination of high (H), medium (M), and low (L).

Fuzzy inference engine is used for output (weight) derivation given the inputs with a fuzzy rule-based system using an aggregation method, i.e., the intersection operation. Figure 8 shows an example of EW fuzzy with 9 rules (See Table 5). There are two normalized inputs, i.e., hop count and remaining energy, here with 0.38 and 0.66 as examples. With these inputs, each fuzzy set is considered with "AND" operation; i.e., the minimum among the twos will be selected to cross over the weight, resulting in output. Next, the aggregation method is applied over the 9 rules before applying the defuzzifier in the next component.

Defuzzifier is used to compute the final output based on the center of gravity (CoG) concept over the derived output (weight). This final output (in the range of 0 to 1) provides the weight for the routing stage; here, 0.577 is an example after the fuzzy inference engine. Note that the output is the energy-assisted routing weight (EW).

Table 6: Example: neighbor table of SN 20 with EW.

Node ID	Hop Count	EW (Weight)
16	4	0.89
17	4	0.75
19	5	0.70
20	5	0.83

(B) EW Derivation with FLS. A weight (EW) will be derived for the FLS (i.e., the final output from Defuzzifier). Then, that weight will be computed once *Request-Stage-Msg* has been received. Subsequently, SN will reply with this weight, in addition to its own identity, via *Reply-Stage-Msg*. With the reply message, the corresponding weight will be stored in the neighbor table. Table 6 shows an example of the update of the 20th SN. Here, this SN broadcasts the request message to its neighbors, SNs 16, 17, 19, and 20. These SNs send the reply to update the base, for example, with weights of 0.89, 0.75, 0.70, and 0.83, respectively. Note that the selection of the next hop will be discussed again in Section 4.4.

4.3. Congestion Control and Prediction. Previously, two main factors, namely, hop count and remaining energy, were used as the main criteria for determining a path. However, in dense deployments, another key factor is network congestion, which typically lowers the overall performance, e.g., through packet loss, low throughput, and high latency. Thus, we also investigated this constraint using a statistical congestion prediction model (in consideration of buffer occupancy) with FLS-based congestion controls. Similar to Section 4.2, however, here, we additionally tuned the membership function of FLS with BAT for the two inputs, namely, predicted buffer occupancy and forwarding rate, to derive another weight factor (Availability Weight (AW)).

4.3.1. Congestion Prediction with Exponential Smoothing

(A) Exponential Smoothing. In this research, we applied exponential smoothing, which is one of the most potent time-series prediction models. Its key advantage over short-term prediction was appropriately used for WSNs [34] to aid in path selection to avoid congestion in consideration of buffer occupancy. Equations (3) to (5) show the exponential smoothing derivation:

$$\widehat{Z}_t(1) = \frac{S_t}{W_t} \tag{3}$$

$$S_t = \sum_{\gamma=0}^{t-1} \alpha^{\gamma} Z_{t-\gamma} \tag{4}$$

$$\widehat{Z}_{t+1} = (1-\alpha) Z_t + \alpha \widehat{Z}_{t-1} \tag{5}$$

where $\widehat{Z}_t$ is the predicted value at time t (buffer), W_t denotes a geometric series, and α is a given constant

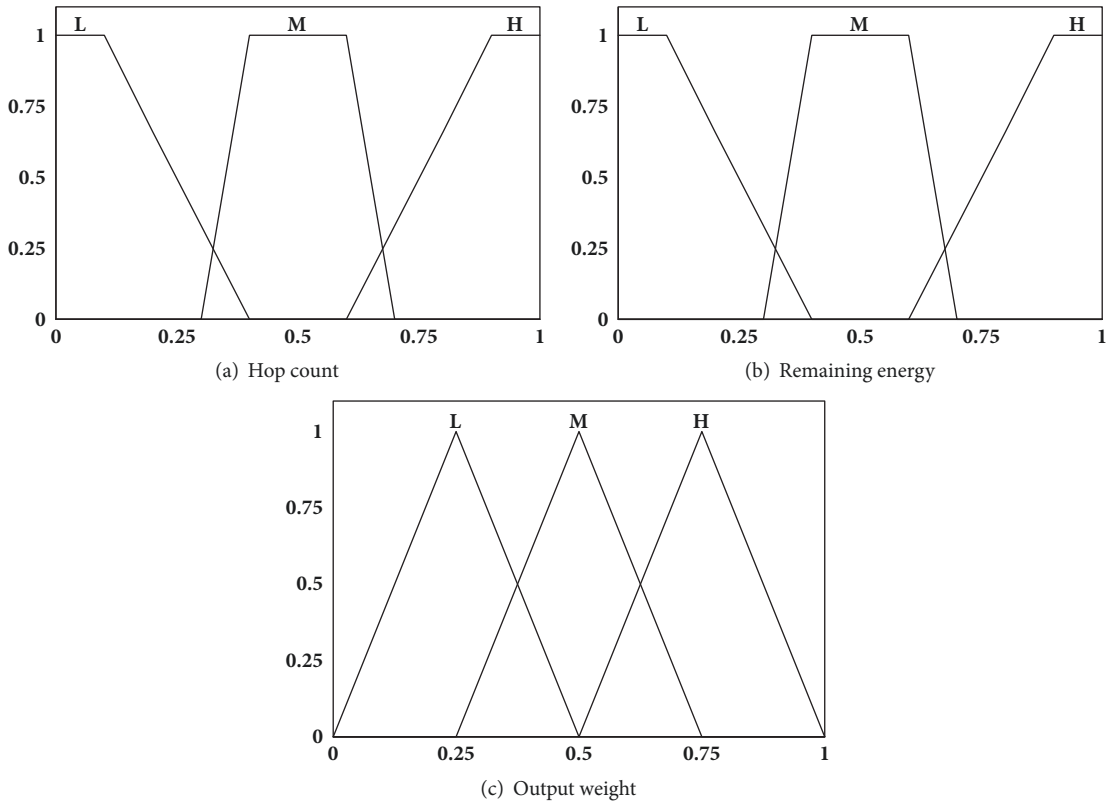


Figure 7: EW membership function: Hop count, remaining energy, and output weight.

(smoothing factor) in the range of 0.02 and 0.3. Equation (6) shows how to derive the predicted buffer, namely, BO over $\widehat{Z}_t$.

$$BO_{t+1} = (1 - \alpha)\, BO_t + (\alpha)\, BO_{t-1} \tag{6}$$

(B) Adaptive Membership Function Using BAT. BAT, introduced by X. S. Yang et al. [35], is a naturally inspired approach used as a metaheuristic algorithm against an NP-complete problem [36] to find a good solution within a constraint-like resource (such as computational time complexity). According to a survey provided by A. H. Gandomi et al. [37], BAT is a potent algorithm in terms of performance compared with particle swarm optimization (PSO) and genetic algorithms (GAs).

BAT is also used for WSNs in several areas, such as localization [38] and radial optimization [39]. Thus, we adopt BAT to tune the membership function of FLS.

In general, BAT imitates bat behavior, in particular, the food scavenging behavior of microbats. A bat generates a high-frequency pulse to navigate toward a food resource; it also informs other bats of the location of the food. There are three main steps:

(i) Each bat sends a pulse (echolocation) to identify the position and distance for food, prey, and obstacle differentiation. The objective function $f(x)$ is defined in terms of the bat population x_i with velocity v_i and transmission frequency f_i to determine a better solution $f(x)$ in each round N. In each round, a better solution will be determined by adjusting the pulse rate (r_i) and loudness (l_i).

(ii) Each bat has no specific pattern for flying toward the food resource at velocity v_i from position x_i. The initial frequency is defined as f_{min}. During flying, wavelength λ and its amplitude or loudness l_i will be adjusted to seek the target. The bat can also change the pulse rate r_i in the range of [0, 1] such that the rate is 0 in the initial pulse. The bat also increases the pulse rate once it is close to the target.

(iii) The loudness l_i is initially set to l_0 (maximum amplitude). During flying, the bat can search for the optimal solution. In contrast to the pulse rate, once the bat is close to the target, the loudness tends to be lowered, with l_{min} as the minimum amplitude.

In this research, we applied the bat behavior to adjust the membership function (MF_i) of FLS (see Algorithm 2). Here, $f(x) = MF_i$. In each round, we select the best MF, i.e., the MF with the maximum Euclidean distance (ED) between a specific MF and an initial MF. Typically, the population (number of bats = n) affects both the computational complexity and the accuracy gain [40]. We intensively investigated this trade-off and observed that, with 5 bats ($x_i = 5$), the accuracy was not significantly different from that with a larger number of bats. We defined an iteration threshold (N_i) as a scope of membership function adjustment to maintain the original shape of the function (either from trapezoidal to trapezoidal

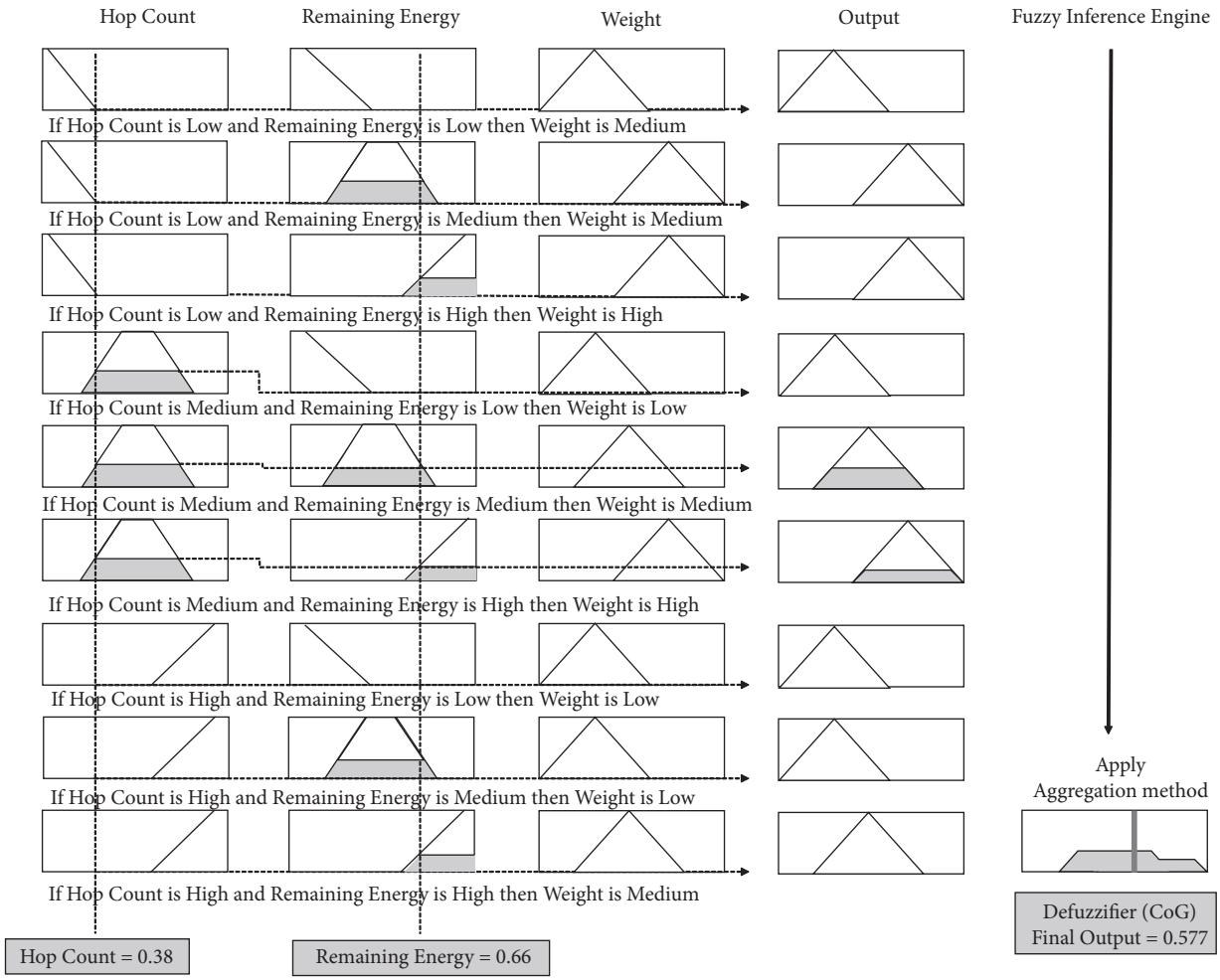


Figure 8: Example: EW fuzzy.

Objective function $f(x) = MF_i$
Initial population size $x_i \mid i \in 1, 2, \ldots, n$
Initial maximum iterations N_i, pulse rate r_i, loudness l_i, velocity v_i.
Define pulse frequency f_i
while (N_i)
Generate new solutions by adjusting f_i
Update vi and MF_i
  if (rand() $\longrightarrow r_i$)
    Select a solution and generate a local solution
  end if
    Generate a new solution with a random flying pattern
  if (rand() $< l_i$ & $f(x_i) < f(x)$)
    Accept new solution MF_i
    Increase r_i and decrease L_i
  end if
  Rank the bats and find the optimal MF_i using ED
end while

Algorithm 2: Membership adjustment using BAT.

Table 7: Example: fuzzy rule of α.

IF (Hop Size)	AND (Hop Count)	THEN (Weight)
L	L	H
L	M	H
L	H	M
M	L	L
M	M	M
M	H	M
H	L	L
H	M	L
H	H	M

Table 8: Example: fuzzy rules of AW.

IF (Buffer Occupancy)	AND (Forwarding Rate)	THEN (Weight)
L	L	L
L	M	H
L	H	VH
M	L	L
M	M	M
M	H	H
H	L	VL
H	M	L
H	H	M

or trapezoidal to triangular; see also Figures 9(a) and 9(c); 9(b) and 9(d) for α as smoothing factor; or Figures 10(a) and 10(c); 10(b) and 10(d) for AW as availability weight). We followed the configuration described by X. S. Yang et al. [35] for r_i, l_i, and f_i.

(C) Fuzzy Logic System with Adaptive Membership Function for Optimized Smoothing Factor. With the exponential smoothing discussed previously, the value of smoothing factor (α) can affect the prediction precision. Thus, we focus on the α derivation with regard to the relationship between hop size (number of hops from the source node) and hop count (number of hops from the sink node) using FLS.

Similar to the previous section (Section 4.2), we applied FLS in addition to membership tuning using BAT in different inputs; here, the two relationships (hop size and hop count) are used as inputs. Note that similar steps are applied.

For example, with Fuzzifier, we examined different membership functions (i.e., the two inputs with a triangular function for low and a trapezoidal function for medium and high) and used a triangular function for the output weight (see Figures 9(a) and 9(b)). Figures 9(c) and 9(d) show the membership after tuning with BAT. Table 7 shows an example with fuzzy rules; here, there are 9 rules.

Note that the same concepts of Fuzzy Inference Engine and Defuzzifier are applied (aggregation method and CoG; see also Figure 8) but with different inputs. Subsequently, the weight after defuzzification is used to determine α as the final output.

From Sections 4.3.1(A) and 4.3.1(B), we applied the optimized α (α_{opt}) for exponential smoothing with buffer as the parameter in a linear relationship, as described in

$$BO_{t+1} = \alpha_{opt} BO_t + \left(1 - \alpha_{opt}\right) BO_{t-1} \quad (7)$$

(D) Fuzzy Logic System with Adaptive Membership Function for Optimized Availability Weight. We considered the exponential smoothing for buffer occupancy (BO) prediction that was derived previously and the forwarding rate (FR) as the two input parameters over FLS for determining the availability weight (AW). Note that BO is used to determine the constraint of packet processing (queuing = drop if buffer is full), and FR is for the transmission capability. Note that these two factors are used to avoid a path that can lead to a high probability of packet loss (due to congestion, which results in queue unavailability and a low transmission rate).

Similar to the previous section (Section 4.2), we applied FLS in addition to membership tuning using BAT in different inputs: the predicted buffer occupancy and the forwarding rate. Similar steps were also applied. For example, with Fuzzifier, we examined different membership functions (i.e., the two inputs with triangular functions for low and high and a trapezoidal function for medium) and a triangular function for the output weight (to achieve high precision for high performance achievement, we used five membership levels instead of three, very low, low, medium, high, and very high; see also Figures 10(a) and 10(b)).

Figures 10(c) and 10(d) show the membership after tuning with BAT. Table 8 shows an example of fuzzy rules; here, there are 9 rules. The same concepts of Fuzzy Inference Engine and Defuzzifier are applied (aggregation method and CoG; see also Figure 8) but with different inputs. Subsequently, the weight after defuzzification is used to determine the AW as the final output.

Figure 11 also shows a detailed sequence diagram of our proposal, OFES.

4.4. Path Determination. As an example, shown in Figure 12, OFES can select the path from source to sink from 3 classes based on the user requirements and the presence of weight information.

(i) **Case I:** Top-Down Hierarchical Structure (Section 4.1). After the Route Create phase, all SNs determine their hop counts from the sink node. To send the packet upward, an SN can simply select the parent with the lowest hop count (stored in the neighbor table) to the sink node. In case there is more than one possible next hop, the preferred next hop is the lowest node identification (this is also applied for the other two cases below).

(ii) **Case II:** Energy-Assisted Routing Weight (Section 4.2). With an additional process for weight determination given the two factors (hop count and remaining energy), a particular SN can select the parent with the maximum EW from the neighbor table.

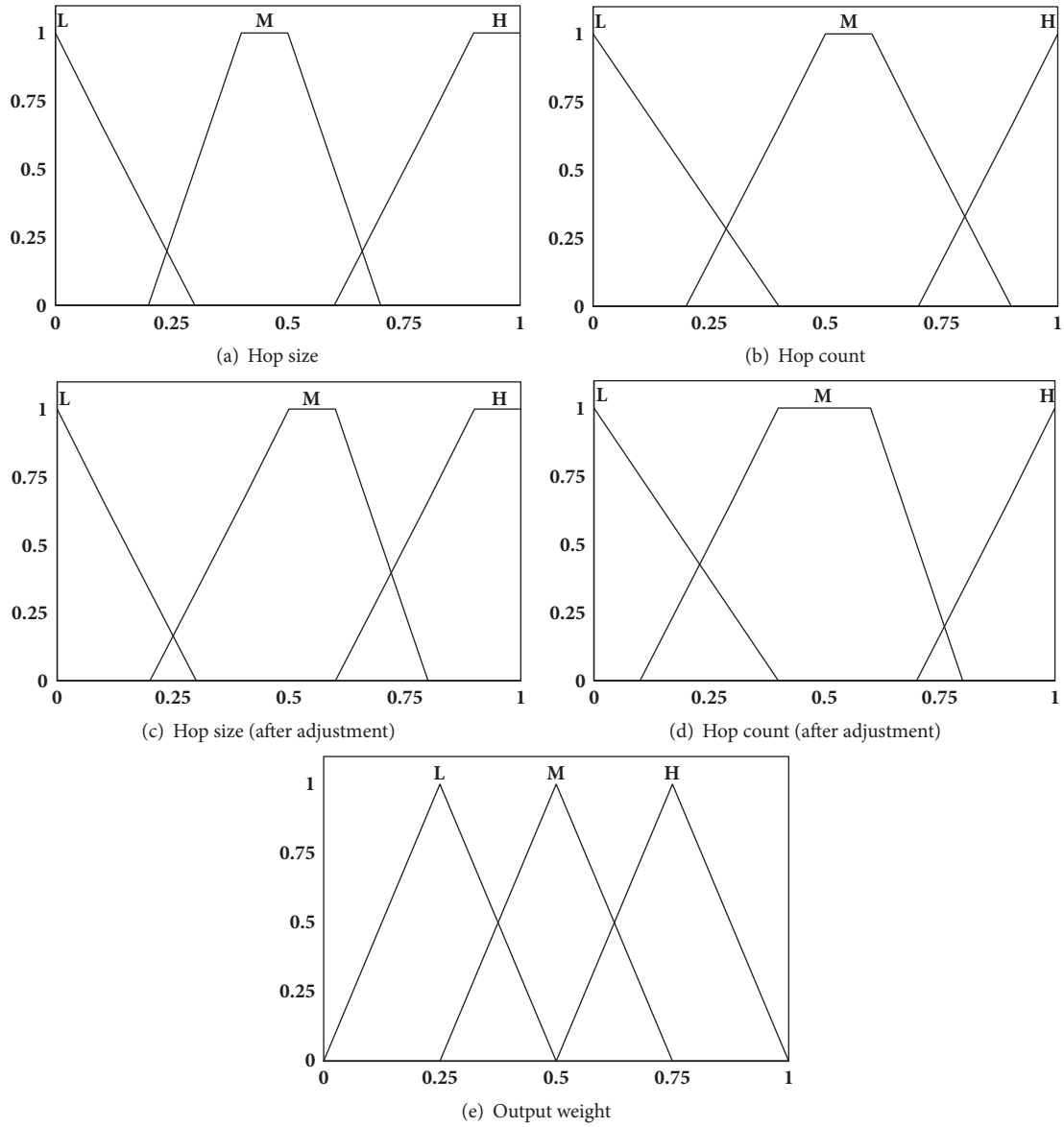


Figure 9: α membership function (before and after BAT tuning): hop count, hop size, and output weight.

(iii) **Case III:** Congestion Control and Prediction (Section 4.3). Similar to the second case, but with an additional weight (AW), here, we defined link score (LS) as a path selection criterion over EW and AW, as shown in (8). Here, γ is a weighted constant for determining a relationship between these two factors, which ranges between 0 and 1:

$$LS = \gamma AW + (1 - \gamma) EW \tag{8}$$

Once LS is determined, a particular SN can select the parent with the maximum LS.

Figure 12 shows an example of path determination with different cases. Here, we only show the way in which the 20th SN selects a path given the neighbor table. Figure 12(a) shows Case I; here, the lowest hop count can be determined from 20 ⟶ 17 ⟶ 13 ⟶ 7 ⟶ 2 ⟶ sink node. However, in the second case, Figure 12(b) shows a different path selection (20 ⟶ 17 ⟶ 13 ⟶ 14 ⟶ 9 ⟶ 4 ⟶ sink node) with an additional weight (EW) (see Table 9). We select the maximum EW path. Figure 12(c) shows a different path (20 ⟶ 16 ⟶ 12 ⟶ 11 ⟶ 6 ⟶ 1 ⟶ sink node) while selecting LS (AW and EW) (see also Table 10). Note that after LS determination, the LSs of SNs 16, 17, and 19 are 0.875, 0.8, and 0.835, respectively. Thus, SN 16 is the parent of SN 20.

Table 9: Example:neighbor table of SN 20 with EW.

Node ID	Hop Count	EW (Weight)
16	4	0.89
17	4	0.65
19	5	0.83

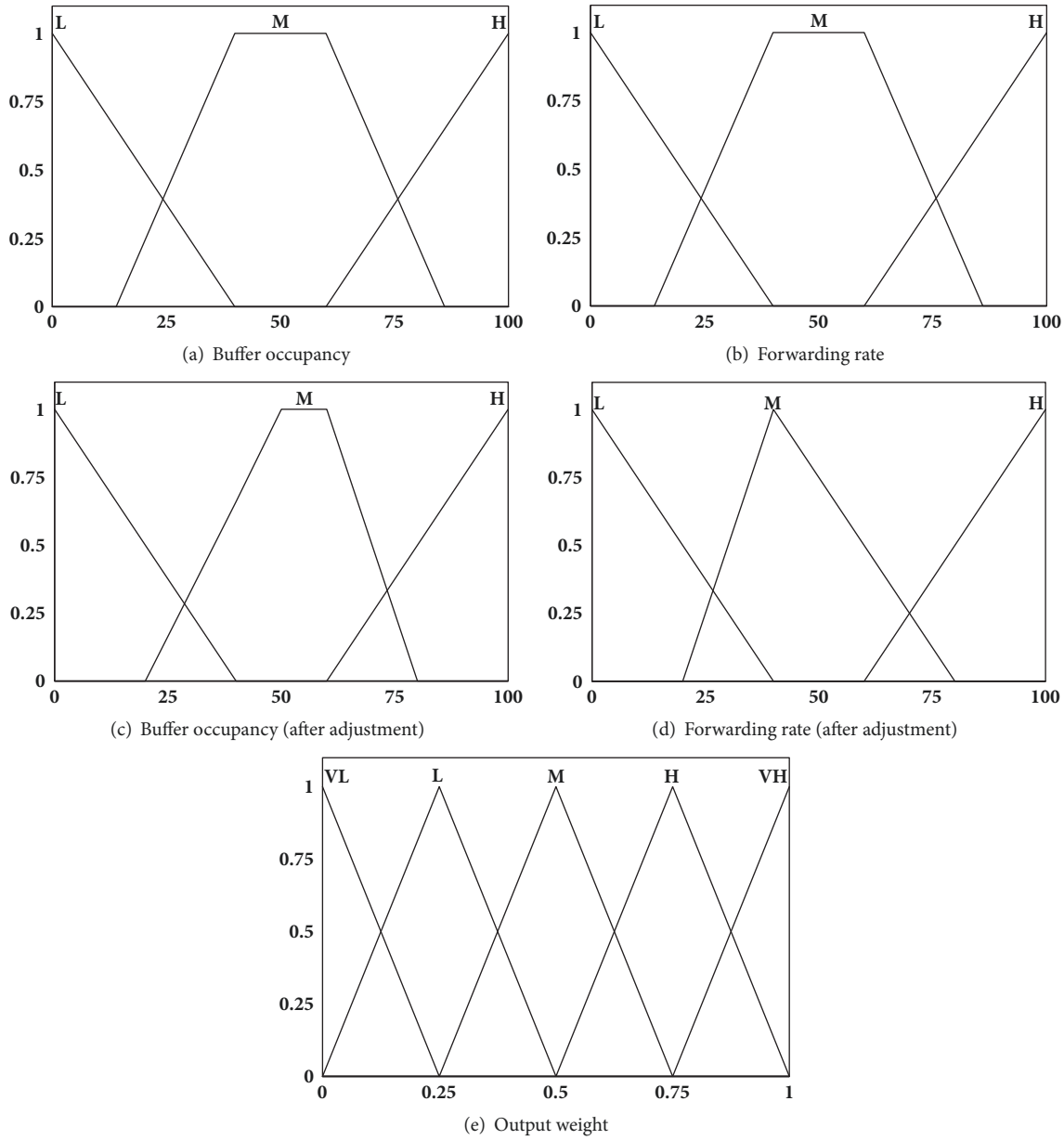


Figure 10: AW membership function (before and after BAT tuning): buffer occupancy, forwarding rate, and output weight.

Table 10: Example: neighbor table of SN 20 with EW and AW.

Node ID	Hop Count	EW (Weight)	AW (Weight)
16	4	0.89	0.86
17	4	0.65	0.95
19	5	0.83	0.84

5. Performance Evaluation

To confirm the superior performance of OFES, we evaluated its performance in comparison with three state-of-the-art methods: HTAP [21], DAIPaS [22], and CCEbH [23], including the traditional ad hoc routing, AODV [31]. In addition, we evaluated our design criteria for the three possible path determinations, including the three cases: Top-Down Hierarchical Structure, Energy-Assisted Routing Weight, and Congestion Control and Prediction (with and without BAT tuning).

5.1. Experimental Design. Our algorithm was simulated using network simulation (NS2) version 2.35 with an extension on WSN [41, 42]. We simulated FLS using Octave 4.2. Our testbed was an Ubuntu 14.04 LTS. The SNs were randomly deployed in a topographical area A of dimension 200 m × 200 m. Once deployed, there was no mobility involved. The number of sensor nodes was 100, with 50 meters as the communication range (see an example shown in Figure 13). Note that the selected parameters follow the specifications

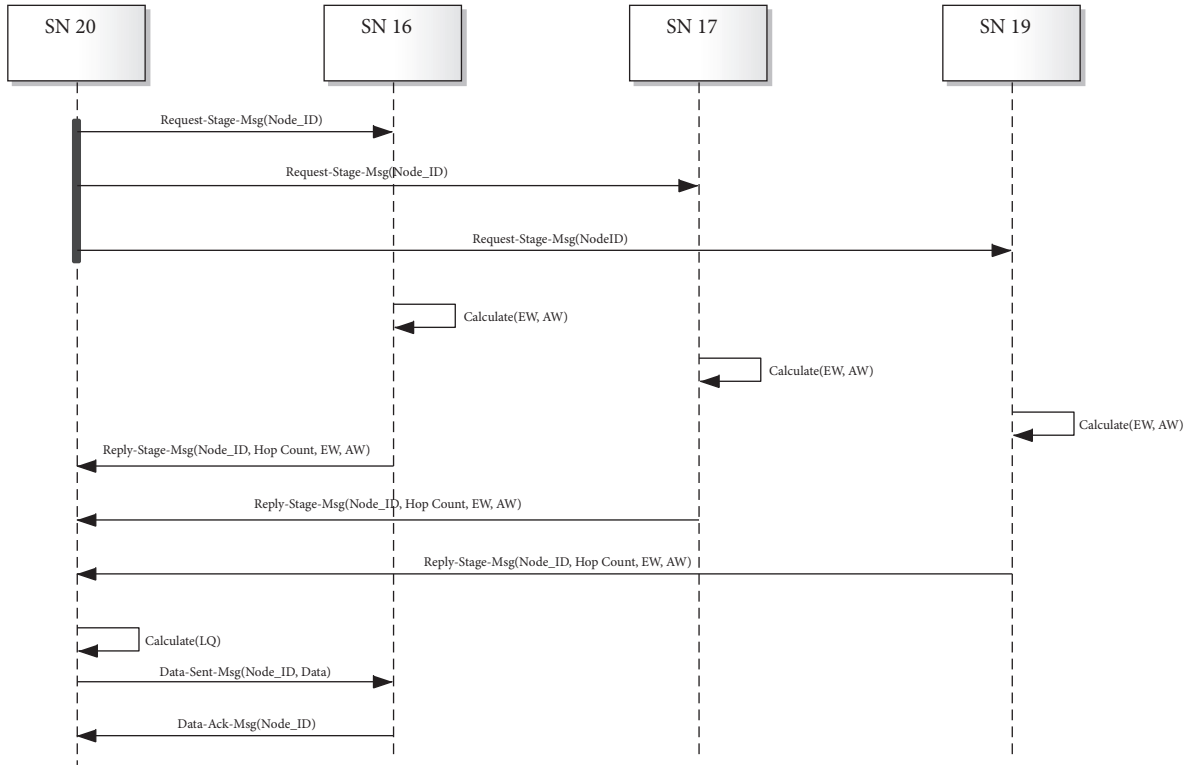


Figure 11: Sequence diagram of OFES.

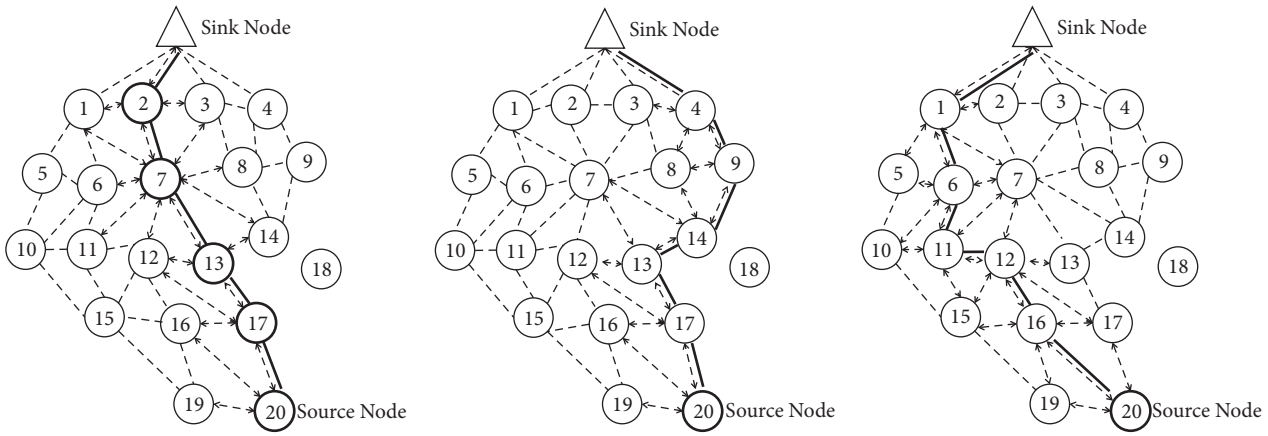


Figure 12: Example: path determination.

of MicaZ [30], HTAP [21], DAIPaS [22], and CCEbH [23].

To evaluate the efficiencies of the methods, experiments were conducted on both sparse and dense networks in terms of the number of events (more traffic), i.e., 10% and 20%, where each is from 32 Kbps to 224 Kbps (with a step size of 32 Kbps). The initial energy was 10 J [21–23]. The simulation was over 300 seconds with 10 trials, and the average result was calculated. The parameter values are listed in Table 11.

We also performed the intensive evaluation over the possible gamma (0 to 1) for optimal use in OFES at the two event rates and found that, at 0.48 and 0.62 in 10% and 20%, the event rate can outperform, i.e., 147 and 124 Kbps (see Figure 14), and we used these findings for further evaluation.

For comparison purposes with [21–23], three main metrics are used in the performance evaluation, namely, Average Throughput at Sink (ATS), Packet Delivery Ratio (PDR),

Table 11: Simulation parameters.

Parameter	Value
Sensor Node (Mote)	Micaz
Initial Energy	10 Joules
Transmission range	50 meters
Maximum bandwidth	250 Kbps
Number of sink node	1
Number of SNs	100
Event Rate	10% and 20%
Link Layer	CSMA/CA
Propagation model	Free Space
Antenna	Omni
Interface queue	Droptail
Packet size	128 bytes
Queue size	100 packets
Area	200 m^2
Simulation time	300 seconds

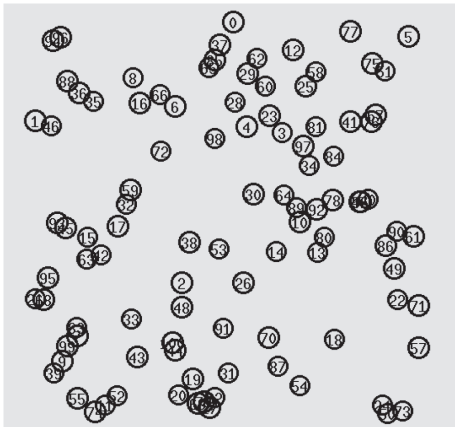


Figure 13: Node distribution.

and Average Remaining Energy (ARE), which are defined as follows:

$$ATS = \frac{\sum_{i=0}^{n} Rx\_sink_i}{\#Event\_Node} \quad (9)$$

$$PDR = \frac{\sum_{i=0}^{n} Packet\ Received}{\sum_{i=0}^{n} Packet\ Sent} \quad (10)$$

$$ARE = \frac{\sum_{i=0}^{n} \left(E_{init(i)} - E_{usage(i)}\right)}{\#SNs} \quad (11)$$

where Rx_sink_i is the received traffic from the ith SN by the sink node and $E_{init(i)}$ and $E_{usage(i)}$ are the initial and usage energies of the ith SN, respectively.

There are two scenarios to justify our performance: (1) we evaluated OFES with our experimental design justification over our three motivations as path determination, Case I (Top-Down Hierarchical Structure, OFES-Case-I), Case II (Energy-Assisted Routing Weight, OFES-Case-II), and Case III (Congestion Control and Prediction with and without BAT tuning, OFES-Case-III-woBAT and OFES-Case-III-withBAT).

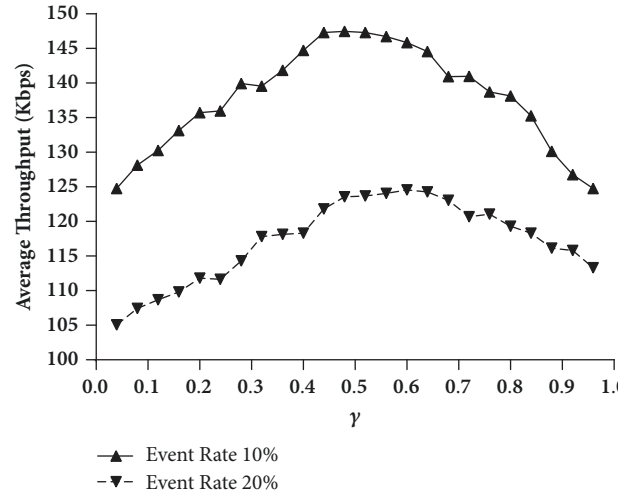


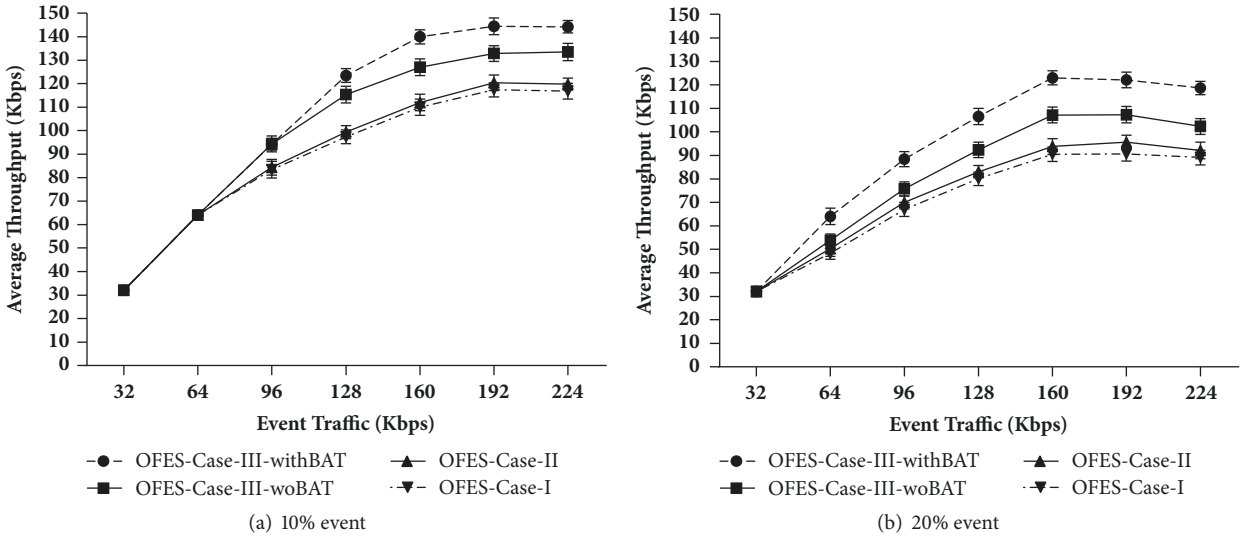
Figure 14: Average throughput over γ.

5.2. Simulation Results and Discussion. For the first scenario, our motivation with regard to the three cases of path determination, Figure 15 shows the performance in terms of ATS (Kbps). In general, as shown in Figure 15(a), our finalized OFES, OFES-Case-III-withBAT, outperforms its three derivations (OFES-Case-III-woBAT, OFES-Case-II, and OFES-Case-I) by factors of 7.25%, 21.61%, and 24.10%, with 10% events (sparse mode), respectively; and 13.28%, 28.44%, and 39.31%, with 20% events (dense mode); although the performance (throughput) is not in such an order. There is no congestion during low traffic, e.g., 32 to 64 Kbps in sparse mode and 32 Kbps in dense mode.

In terms of PDR, Figure 16 shows the results corresponding to ATS; however, the results are the opposite: the higher the traffic, the lower the PDR, for example, from 100% to 66%, 59%, 53%, and 52% in sparse mode and to 48%, 45%, 41%, and 39% in dense mode, for the four cases, respectively. Our finalized OFES, Case-III-withBAT, is superior on average to its derivatives.

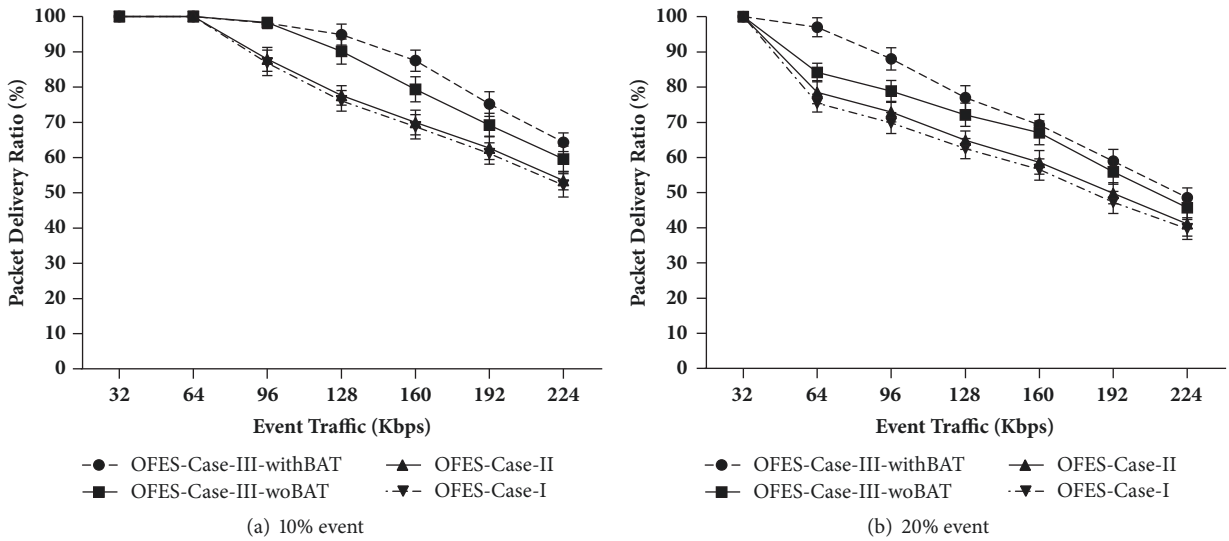
Figure 17 shows the ARE values of our OFESs. In general, with more traffic, the remaining energy will decrease. For example, from 32 Kbps to 224 Kbps, traffic decreases the energy from 94% to 53%, 91% to 45%, 89% to 37%, and 85% to 32% in sparse mode and 86% to 27%, 82% to 20%, 80% to 16%, and 79% to 11% in dense mode, respectively. Our finalized OFES (OFES-Case-III-withBAT) is superior, and it outperforms the other three derivatives by 10.83%, 23.61%, and 41.39% in sparse mode and 13.31%, 32.91%, and 55.37% in dense mode.

For the second scenario, compared with the recent congestion control algorithms in WSNs, Figure 18 shows the performance in terms of ATS (Kbps). In general, with the increase of event traffic, the average throughput decreases due to network congestion, for example, from 32 Kbps to 147, 133, 126, 119, and 92 Kbps, with 10% events (sparse mode), for OFES (aka OFES-Case-III-withBAT), CCEbH, DAIPaS,



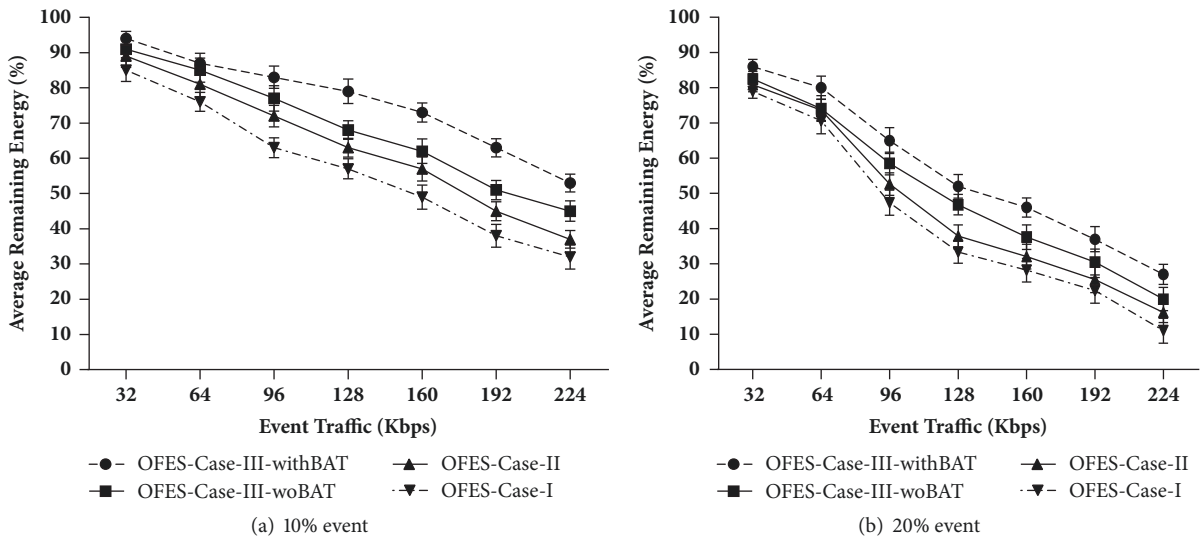
(a) 10% event (b) 20% event

Figure 15: Average throughputs at sink node over different traffic loads.



(a) 10% event (b) 20% event

Figure 16: Packet delivery ratios over different traffic loads.



(a) 10% event (b) 20% event

Figure 17: Average remaining energies over different traffic loads of OFESs (OFES-Case-I, OFES-Case-II, OFES-CaseIII-woBAT, and OFES-Case-III-withBAT).

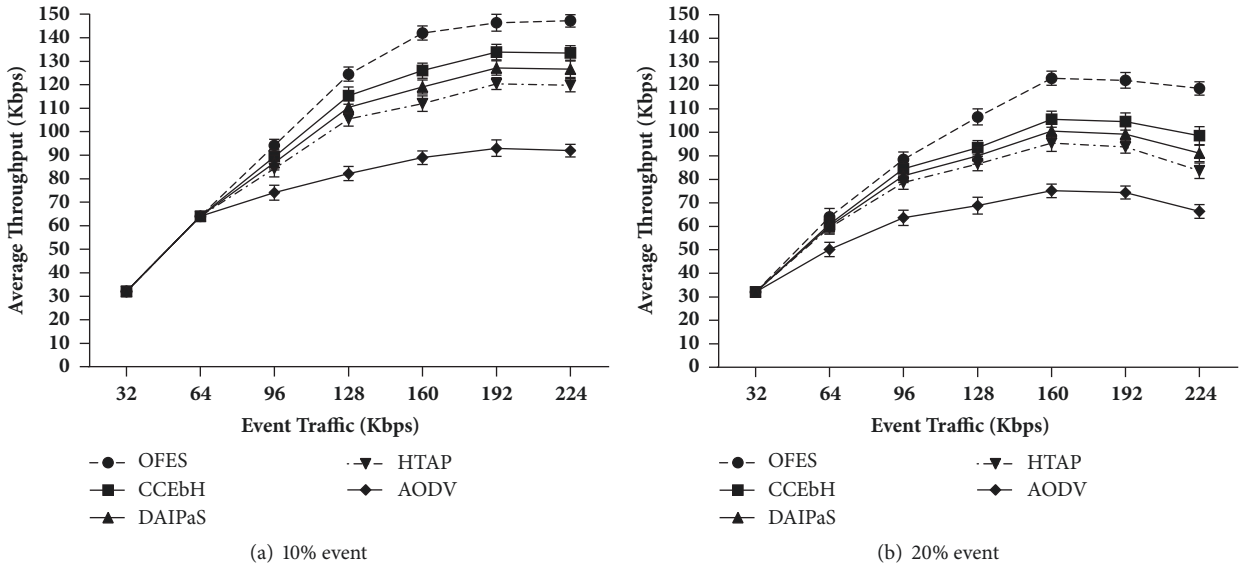


Figure 18: Average throughputs at sink node over different traffic loads.

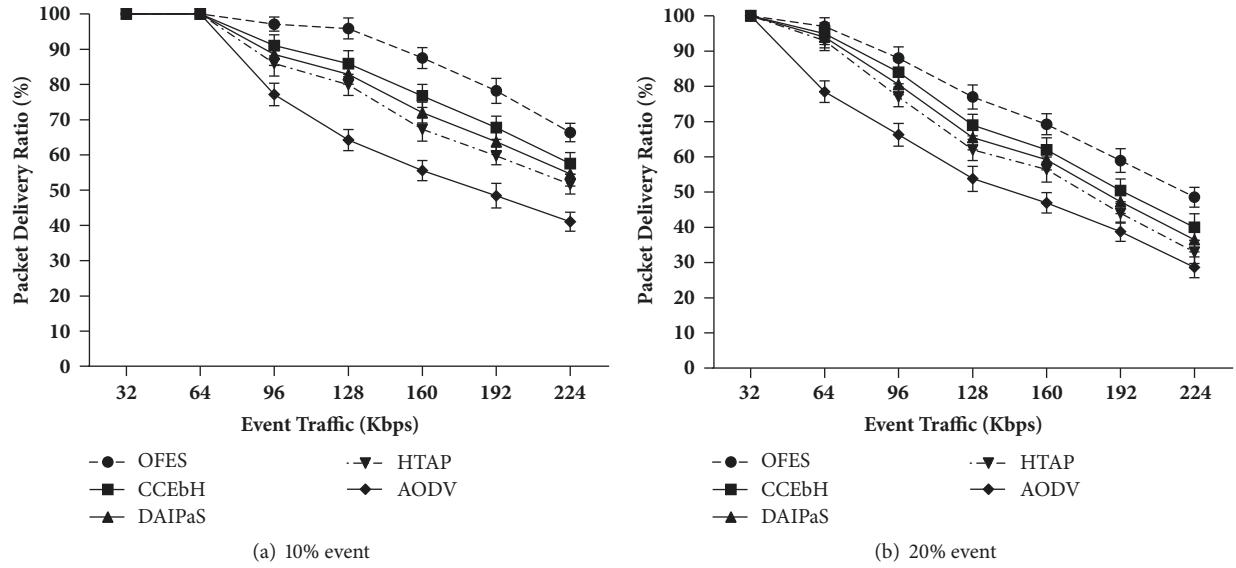


Figure 19: Packet delivery ratios over different traffic loads.

HTAP, and AODV, respectively; and from 32 Kbps to 118, 98, 91, 83, and 66 Kbps, with 20% events (dense mode).

In general, our proposed method outperforms the other four considered methods by factors of 8.32%, 14.37%, 20.18%, and 50.10% higher than the others. Note that, with low traffic, i.e., 32 and 64 Kbps, there is no congestion involved, so there is no loss (ATS is the same as the event traffic); however, with high traffic, the loss becomes obvious (low ATS on average). In dense mode, for 20% events, the ATS was similar despite the increased congestion. Thus, ATS tends to be lower in the dense mode than in the sparse mode. However, OFES is still superior, in particular, at 160 Kbps with ATS = 123 Kbps, and, on average, it outperforms the other four methods by 11.22%, 18.19%, 24.04%, and 54.70%. Again, note that there is no congestion at 32 Kbps traffic.

In terms of PDR, Figure 19 shows the results corresponding to ATS but in the opposite direction: with an increase in event traffic, PDR tends to decrease due to network congestion, for example, from 100% to approximately 66%, 57%, 54%, 51%, and 40%, with 10% events; and 48%, 40%, 36%, 33%, and 28%, with 20% events for OFES, CCEbH, DAIPaS, HTAP, and AODV, respectively. OFES is again superior on average to the other methods. Again, the network becomes congested at approximately 64 Kbps with 10% events. Similarly, in dense mode, where the number of events is 20%, PDR becomes much lower due to high congestion.

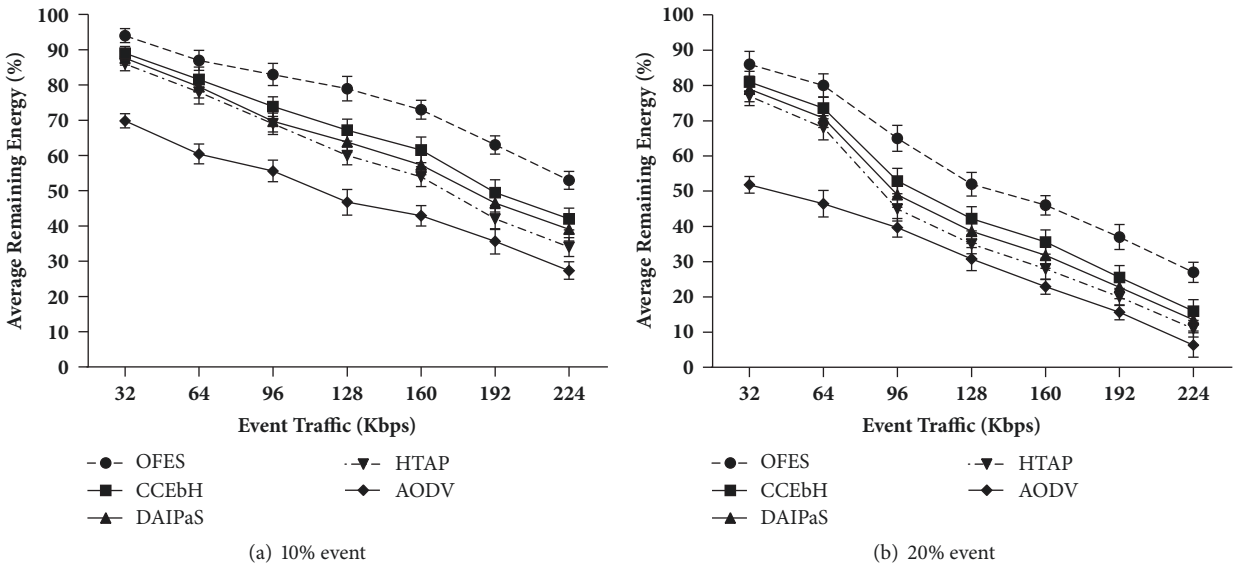


Figure 20: Average remaining energies over different traffic loads of OFES, CCEbH, DAIPaS, HTAP, and AODV.

Figure 20 shows ARE values of the three protocols. In general, as shown in Figure 20(a), with more traffic, the remaining energy will decrease, e.g., from 32 Kbps to 224 Kbps: from 94% to 53%, 89% to 42%, 87% to 39%, 86% to 34%, and 69% to 27%, respectively. Again, OFES is superior, and the overall remaining energy is high. Figure 20(b) also shows the results for the dense mode. OFES remains superior and outperforms CCEbH, DAIPaS, HTAP, and AODV by 13.62%, 22.56%, 30.55%, and 62.48%, respectively (20.81%, 42.37%, 59.58, and 82.27% for dense mode).

6. Conclusion and Future Work

In this research, we propose a path determination architecture for WSNs that takes into account network congestion, which is called the Optimized Fuzzy Logic-based Congestion Control Scheme with Exponential Smoothing Prediction in WSNs (OFES). For flexible use, three main path determination approaches can be applied based on the available resources and information: (1) a simplified level-based hierarchical tree path construction (using a hop count to find a path), (2) a more complex path determination method that uses known information, such as hop count and remaining energy, and (3) a path resolver with congestion prediction that uses exponential smoothing and two factors, such as buffer occupancy and forwarding rate. We also applied FLS to the last method to determine an adjustable weight in addition to weight optimization using BAT to tune the membership function.

To justify our motivation of path determination and to evaluate the performance of OFES, we first compared OFES with the other three cases, i.e., Case I (Top-Down Hierarchical Structure), Case II (Energy-Assisted Routing Weight), and Case III (Congestion Control and Prediction with and without BAT tuning). The results justify our design such that our finalized OFES (Case III with BAT tuning) is superior. Then, we compared the OFES with the three recently proposed methods that include the traditional ad hoc routing, namely, CCEbH, DAIPaS, HTAP, and AODV in terms of throughput, packet loss, and energy consumption. Through simulation experiments, we showed that OFES is superior in terms of all metrics, followed by CCEbH, DAIPaS, HTAP, and AODV.

For example, OFES outperformed the other four methods by factors of 9.77%, 16.28%, 22.11%, and 52.4%, respectively, in terms of average throughput and packet delivery ratio; and factors of 17.21%, 32.46%, 45.06% and 72.37%, respectively, in terms of average remaining energy. Note that although the performance of OFES is high, limitations and considerations for further investigation remain, such as the high volume of data transmission. In future work, comprehensive simulation and analysis could be carried out to investigate properties such as network density and diversity, network dimension, mobility, various signal propagation models, network failure probability, and heterogeneous data traffic, including physical device implementation.

Data Availability

No data were used to support this study.

Conflicts of Interest

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

Acknowledgments

This work was supported by grants from the Research Fund for Supporting Lecturers to Admit High Potential Students to Study and Research in Their Expert Program Year 2017 from the Graduate School, Khon Kaen University, Thailand

(Grant no. 601T213), and partially supported by the Thailand Research Fund (TRF) under Grant no. RTA6080013.

References

[1] Gartner. Press Release - Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016, 2017. http://www.gartner.com/newsroom/id/3598917.

[2] Intel. A Guide to the Internet of Things Infographic, 2017. https://www.intel.com/content/www/us/en/internet-of-things/infographics/guide-to-iot.html.

[3] ABI Research, "Big Data and Analytics in IoT and M2M, 2016," https://www.abiresearch.com/market-research/product/1024282-big-data-and-analytics-in-iot-and-m2m/.

[4] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 553–576, 2016.

[5] M. A. Mahmood, W. K. G. Seah, and I. Welch, "Reliability in wireless sensor networks: a survey and challenges ahead," *Computer Networks*, vol. 79, pp. 166–187, 2015.

[6] T. G. Nguyen, C. So-In, and N. Nguyen, "Barrier Coverage Deployment Algorithms for Mobile Sensor Networks," *Journal of Internet Technology*, vol. 18, no. 7, pp. 1689–1699, December 2017.

[7] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 828–854, 2017.

[8] D. Chen, Z. Liu, L. Wang, M. Dou, J. Chen, and H. Li, "Natural disaster monitoring with wireless sensor networks: A case study of data-intensive applications upon low-cost scalable systems," *Mobile Networks and Applications*, vol. 18, no. 5, pp. 651–663, 2013.

[9] S. Halder and S. D. Bit, "Enhancement of wireless sensor network lifetime by deploying heterogeneous nodes," *Journal of Network and Computer Applications*, vol. 38, no. 1, pp. 106–124, 2014.

[10] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel, "Topology management techniques for tolerating node failures in wireless sensor networks: a survey," *Computer Networks*, vol. 58, no. 1, pp. 254–283, 2014.

[11] M. A. Kafi, D. Djenouri, J. Ben-Othman, and N. Badache, "Congestion control protocols in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1369–1390, 2014.

[12] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications*, vol. 60, pp. 192–219, 2016.

[13] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[14] S.-U. Lar and X. Liao, "An initiative for a classified bibliography on TCP/IP congestion control," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 126–133, 2013.

[15] C. Lochert, B. Scheuermann, and M. Mauve, "A survey on congestion control for mobile ad hoc networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 5, pp. 655–676, 2007.

[16] J. A. Khan, H. K. Qureshi, and A. Iqbal, "Energy management in Wireless Sensor Networks: A survey," *Computers & Electrical Engineering*, vol. 41, pp. 159–176, 2015.

[17] C. Sergiou, P. Antoniou, and V. Vassiliou, "A comprehensive survey of congestion control protocols in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1839–1859, 2014.

[18] S. A. Shah, B. Nazir, and I. A. Khan, "Congestion control algorithms in wireless sensor networks: Trends and opportunities," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 3, pp. 236–245, 2017.

[19] A. Ghaffari, "Congestion control mechanisms in wireless sensor networks: a survey," *Journal of Network and Computer Applications*, vol. 52, pp. 101–115, 2015.

[20] J. Kang, Y. Zhang, and B. Nath, "TARA: Topology-aware resource adaptation to alleviate congestion in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, pp. 919–931, 2007.

[21] C. Sergiou, V. Vassiliou, and A. Paphitis, "Hierarchical Tree Alternative Path (HTAP) algorithm for congestion control in wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 1, pp. 257–272, 2013.

[22] C. Sergiou, V. Vassiliou, and A. Paphitis, "Congestion control in Wireless Sensor Networks through dynamic alternative path selection," *Computer Networks*, vol. 75, pp. 226–238, 2014.

[23] W. Chen, Y. Niu, and Y. Zou, "Congestion control and energy-balanced scheme based on the hierarchy for WSNs," *IET Wireless Sensor Systems*, vol. 7, no. 1, pp. 1–8, 2017.

[24] S. Jaiswal and A. Yadav, "Fuzzy based adaptive congestion control in wireless sensor networks," in *Proceedings of the 2013 Sixth International Conference on Contemporary Computing (IC3)*, pp. 433–438, Noida, India, August 2013.

[25] Y. Zhang, J. Wang, D. Han, H. Wu, and R. Zhou, "Fuzzy-Logic Based Distributed Energy-Efficient Clustering Algorithm for Wireless Sensor Networks," *Sensors*, vol. 17, no. 7, pp. 1554:1–1554:21, 2017.

[26] H. D. Nikokheslat and A. Ghaffari, "Protocol for Controlling Congestion in Wireless Sensor Networks," *Wireless Personal Communications*, vol. 95, no. 3, pp. 3233–3251, 2017.

[27] H. Z. Shi, R. V. Prasad, E. Onur, and I. G. M. M. Niemegeers, "Fairness in wireless networks: issues, measures and challenges," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 5–24, 2014.

[28] A. Uthra Rajan, S. V. Kasmir Raja, A. Jeyasekar, and A. J. Lattanze, "Energy-efficient predictive congestion control for wireless sensor networks," *IET Wireless Sensor Systems*, vol. 5, no. 3, pp. 115–123, 2015.

[29] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: a top-down survey," *Computer Networks*, vol. 67, pp. 104–122, 2014.

[30] Crossbow technology inc., "MiCaZ Datasheet," Document Part No. 6020-0060-04, January, 2006.

[31] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," No. RFC 3561, 2003.

[32] G. Han, L. Liu, J. Jiang, L. Shu, and G. Hancke, "Analysis of Energy-Efficient Connected Target Coverage Algorithms for Industrial Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 135–143, December 2017.

[33] P. Nayak and B. Vathasavai, "Energy Efficient Clustering Algorithm for Multi-Hop Wireless Sensor Network Using Type-2 Fuzzy Logic," *IEEE Sensors Journal*, vol. 17, no. 14, pp. 4492–4499, 2017.

[34] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with exponential smoothing: the state space approach*, Springer Science & Business Media, 2008.

[35] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization*, vol. 248, pp. 65–74, 2010.

[36] X.-S. Yang and A. H. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.

[37] A. H. Gandomi, X. S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.

[38] C. So-In and W. Katekaew, "Hybrid Fuzzy Centroid with MDV-Hop BAT Localization Algorithms in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, article 894560, 2015.

[39] S. P. Kaur and M. Sharma, "Radially optimized zone-divided energy-aware Wireless Sensor Networks (WSN) protocol using BA (Bat Algorithm)," *IETE Journal of Research*, vol. 61, no. 2, pp. 170–179, 2015.

[40] S. Yilmaz and E. U. Küçüksille, "A new modification approach on bat algorithm for solving optimization problems," *Applied Soft Computing*, vol. 28, pp. 259–275, 2015.

[41] K. Fall and K. Varadhan, "The network simulator (ns-2), 2007," http://www.isi.edu/nsnam/ns.

[42] Manna Research Group, "Mannasim framework, 2010," http://www.mannasim.dcc.ufmg.br/index.htm.