

FEAR: Fuzzy-Based Energy Aware Routing Protocol for Wireless Sensor Networks

Iman M. ALMomani, Maha K. Saadeh

*Computer Science department, King Abdullah II School for Information Technology (KASIT),
The University of Jordan, Amman, Jordan*

E-mail: i.momani@ju.edu.jo, saadeh.maha@yahoo.com

Received April 9, 2011; revised April 28, 2011; accepted May 7, 2011

Abstract

Wireless Sensor Networks (WSNs) are used in different civilian, military, and industrial applications. Recently, many routing protocols have been proposed attempting to find suitable routes to transmit data. In this paper we propose a Fuzzy Energy Aware tree-based Routing (FEAR) protocol that aims to enhance existing tree-based routing protocols and prolong the network's life time by considering sensors' limited energy. The design and implementation of the new protocol is based on cross-layer structure where information from different layers are utilized to achieve the best power saving. Each node maintains a list of its neighbors in order to use neighbors' links in addition to the parent-child links. The protocol is tested and compared with other tree-based protocols and the simulation results show that FEAR protocol is more energy-efficient than comparable protocols. According to the results FEAR protocol saves up to 70.5% in the number of generated control messages and up to 55.08% in the consumed power.

Keywords: Energy Awareness, Fuzzy Logic, Routing Protocol, Wireless Sensor Network, WSN

1. Introduction

WSNs are envisaged to become a very significant enabling technology in many sectors as they are widely used in many civilian and military as well as industrial applications [1,2]. Unlike traditional wireless communication networks, WSN has its own characteristics. It consists of small, low cost, and low power sensors. Each sensor is embedded with a microprocessor and a wireless transceiver to provide data processing and communication capabilities besides its sensing facility.

The significant interest in WSNs comes from its importance in many applications. In the healthcare sector, for example, WSNs are used in different applications such as patients monitoring, diseases diagnosis and management, and elderly people homecare [2-5]. The use of WSNs in healthcare applications aims to provide remote healthcare monitoring by treating and following up patients in their own homes. This out-of-hospital treatment does not replace the role of hospitals and healthcare centers; it rather involves the patients to be active participants in their own healthcare.

As in medical applications, WSNs have a significant role in military applications [6,7]. They can be used in

battlefield monitoring, intelligent guiding, and remote sensing to sense chemical weapons and detect enemies' attacks. In industrial applications WSNs are successfully used to monitor manufacturing processes such as products quality and monitoring equipments status [2,8].

In a WSN sensor nodes are scattered in the sensing field forming a network according to nodes connectivity. Depending on the underlying network topology sensors collect different environmental data and send the observations to the Base Station (BS) which is also known as the sink node. The sink node is responsible for data gathering and processing. Also, it connects the WSN to the Internet or to other networks. Usually the sink node has unconstrained capabilities in terms of data processing, data storage, and power resources. Unlike the sink node, sensors are resource-constrained devices, they have limited processing and storage capacities, finite battery power, and short radio range [1,9].

In addition to devices' limitations, WSNs suffer from different challenges such as multi node-to-node transmission, data redundancy and high unreliability since sensors are subject to physical damage and failure. These limitations and problems present many challenges in the design and deployment of WSNs. Many recent researches

have been carried out attempting to solve the design and implementation issues in WSNs. Some researches attempt to address the problem of limited energy by proposing energy-efficient solutions to prolong the network's lifetime. These solutions include data routing protocols [10,11]. Other researches focus on the problem of unreliable data transmission and provide solutions to increase the network's reliability and data availability such as multi-path and multi-sink protocols [12,13].

This paper proposes a Fuzzy Energy Aware Routing (FEAR) protocol based on tree routing. The protocol aims to enhance the existing Tree-based Routing (TR) protocol supported by IEEE 802.14.5 [14] in terms of reducing the number of hops and solving the problem of node/link failure. It aims to eliminate the need for additional control messages, as in Plus-Tree Routing (PTR) protocol [15], to build the neighbors' tables by exploiting the control messages that are used for the purpose of tree construction. In this way control messages are used to construct the tree and at the same time to build the neighbors' tables, consequently, the network's overhead is reduced. FEAR protocol consists of several phases: tree construction, data transmission and tree reconstruction due to either node(s) or link(s) failure.

The rest of this paper is organized as follows: Section 2 summarizes some related work. In Sections 3 the proposed FEAR protocol is discussed in details and an analytical evaluation for this protocol is also presented. Section 4 explains the fuzzy system used by FEAR protocol. Section 5 discusses the simulation results. Section 6 concludes the paper and presents avenues for future work.

2. Related Work

Many routing protocols have been proposed for WSNs. Some protocols utilize the concept of hierarchical routing to perform energy-efficient routing in WSNs and extend system's lifetime. In these protocols high-energy nodes can be used to process and send information, while low-energy nodes can be used to perform sensing [9,16].

A well-known hierarchical protocol is called Low-Energy Adaptive Clustering Hierarchy (LEACH) [17]. The idea is to divide the network into clusters and choose a cluster head for each one of them. All local cluster heads will be used as routers to the sink. This clustering will save energy since all data is processed locally inside the cluster. Moreover, all transmissions are performed through cluster heads rather than involving all sensor nodes. Cluster heads are changed randomly over time in order to balance the energy dissipation of nodes. However, LEACH is not applicable to networks deployed in large regions, since it uses single-hop routing where each node can transmit data directly to the cluster head.

Another cluster-based routing protocol that can be used for large WSNs is Energy Clustering Protocol (ECP) as proposed in [18]. Unlike LEACH, ECP uses multi-hop routing. It also utilizes nodes on the cluster edge to send data to the neighboring cluster.

Other cluster-based routing protocols were also proposed in [19,20]. Generally, in cluster-based routing protocols the idea of dynamic clustering brings extra overhead, which may diminish the gain in energy consumption.

Other protocols had been presented in the literature are based on constructing a tree between network nodes and use tree routing for data transmission. Tree-based Routing (TR) is one of these protocols that are supported by IEEE 802.15.4 [14]. TR protocol suites small memory, low power and low complexity networks with lightweight nodes and it aims to eliminate the overhead of path searching and updating, therefore, it reduces extensive messages that are exchanged between network nodes. Although this protocol works well in terms of energy saving, it suffers from two drawbacks. First, message transmission depends on source depth; the deeper the node, the longer the path. Second, it suffers from node/link failure that causes nodes isolation. To overcome these drawbacks, many protocols have been proposed to enhance TR performance. Authors in [15] [21-23] attempted to enhance TR by finding a shorter path to be used in data transmission by exploiting neighbor links rather than using only parent-child links. However, these protocols do not consider nodes power in their solutions.

Due to the dynamicity of clustering process and the complexity of optimizing the number of clusters that may reduce the gain in power consumption in cluster-based routing, the proposed FEAR protocol will be based on tree routing. Network energy will be considered during different stages of the proposed FEAR protocol. FEAR also enhances TR protocols by avoiding their shortcomings in terms of solving the node(s)/link(s) failure problem and decreasing the number of required hops to transmit data messages to the sink. FEAR will be compared with both TR [14] and Plus-Tree Routing (PTR) protocol [15]. These two protocols were chosen for comparison as they construct the tree topology with the minimum cost. In addition, PTR protocol provides solutions to recover from node(s)/link(s) failure.

3. FEAR Protocol

The proposed Fuzzy Energy Aware tree-based Routing (FEAR) protocol is a cross-layer protocol. FEAR protocol consists of several phases. Firstly, the protocol constructs a logical tree between network nodes. During the

construction each node will get a logical ID and construct its neighbors table. Secondly, data packets are transmitted using neighbor links in addition to the parent-child links. During this stage both intermediate nodes energy and depth will be considered. Finally, the tree, or part of it, may be reconstructed in the event of either node(s) or link(s) failure or due to a new node entrance.

During both tree construction and tree reconstruction stages, FEAR protocol uses a ranking system based on fuzzy inference so that nodes rank their neighbors. By using fuzzy ranking inference, nodes energy and depth is kept balanced among all sensor nodes. The fuzzy ranking system structure will be discussed in details in the next section.

3.1. Control Messages

Different control messages are defined by this protocol. These messages are listed in **Table 1**. This table shows the types of these messages, their structures, when they are

sent and the actions to be taken when they are received.

3.2. Network Model

The network model of FEAR protocol is described as follows:

- The sensors are scattered in the network field without isolation.
- All sensor nodes have the same capabilities, the same transmission range, and limited power resources.
- Symmetric model is assumed. That means if sensor A is within sensor B's transmission range then sensor B is within sensor A's transmission range.
- All sensors sense data and transmit it to the sink for processing.
- The sink node assumed to have unconstraint resources.
- All sensor nodes are located in fixed places without mobility.

Table 1. FEAR protocol control messages.

Message Type	When to be Sent	Actions By Receivers	Message Structure
Ready	Is sent when: 1. The node gets an ID, so it is used to broadcast the ID and tell other nodes that it is ready to accept children. 2. The node receives a New Node or Request Parent messages.	1. Store the information in the neighbors table. 2. If receiver node does not have ID, it will find neighbors' rank and send Engagement message.	1. Node ID 2. Node power 3. Rank average
Unready	It is used only to broadcast the ID.	Store ID in the neighbors table.	1. Node ID 2. Node power
Engagement	Is sent when receiving a Ready message and used to request for ID and request for Parent.	May send Engagement-Acceptance message.	
Engagement-Acceptance	Is sent as a reply to an Engagement message.	1. Refresh neighbors table. 2. Calculate the ID. 3. Calculate the fuzzy ranking average for its neighbors. 4. Send Ready message.	1. Node ID 2. Node power 3. Offered ID
New Node	Is sent when new node wants to join the network.	Send Ready message or Unready message.	
Request Parent	Is sent when a node cannot reach its parent.	Send Ready message or Unready message.	
Inform	Is sent to tell the neighbors that the node will go down.	Reconstruct the tree according to their relation with the dead node.	Node ID
Change ID	Is sent when any node change its ID due to some failure to tell other nodes to modify the ID in their tables.	1. Modify the ID in neighbors table. 2. If the receiver is one of sender's children it will update its own ID and send Change ID message.	1. Node ID 2. Node Power

3.3. FEAR Protocol Stages

Different stages of FEAR protocol will be discussed in details through the next subsections.

3.3.1. Stage 1: Tree Construction

A sink-rooted tree should be constructed between network nodes before sending a message to the sink. A new addressing scheme is used in this stage to assign a logical ID for each node. Each node uses the assigned ID to calculate its depth and its neighbors' depth. When a node receives an Engagement-Acceptance message, it will calculate its own ID as follows:

$$\text{Node ID} = \text{Parent ID} \parallel \text{Offered ID}$$

Each Offered ID is represented by m digits which are required to represent C_{\max} nodes, where C_{\max} is the maximum number of children a node can have. For example, if $C_{\max} < 10$ then we need only one digit to represent the offered ID (0...9), but for large networks if $C_{\max} < 100$ then we need 2 digits (00...99), and so on. Using this addressing scheme, each node will be able to know the depth for a particular ID.

To construct the tree we assume that all nodes, initially, do not have IDs and they have full energy. Parents can have at most C_{\max} children, and at the end of this stage each node is assigned an ID from which it can calculate its depth.

First, the sink node will set its rank average to 1 which is the maximum rank then the tree construction is started by broadcasting a Ready message to its neighbors, this message contains the sink's ID (sink ID = initial ID), sink's energy, and the average rank. Each node receiving this message will store the sink's information in its neighbors table and after a short period of time it will send an Engagement message to the sink. The Engagement message has two purposes: request for a parent, and request for an ID. For each Engagement message, the sink node will reply by sending an Engagement-Acceptance message only if the number of its children is less than C_{\max} .

When the node gets the Offered ID, it calculates its ID and the fuzzy rank average for its neighbors then it broadcasts a Ready message allowing its neighbors to send Engagement messages. Note that the Engagement messages are sent after a short period during which they can receive Ready messages from other nodes and find the fuzzy rank for each one. This waiting will force the node to be associated with the best possible node among others (the node that has the best fuzzy rank). Thus, a balance is kept between nodes. This process continues until all nodes get IDs and no more Ready messages are sent. **Table 2** illustrates neighbor table structure and **Table 3** illustrates the steps of this phase. **Figure 1** shows

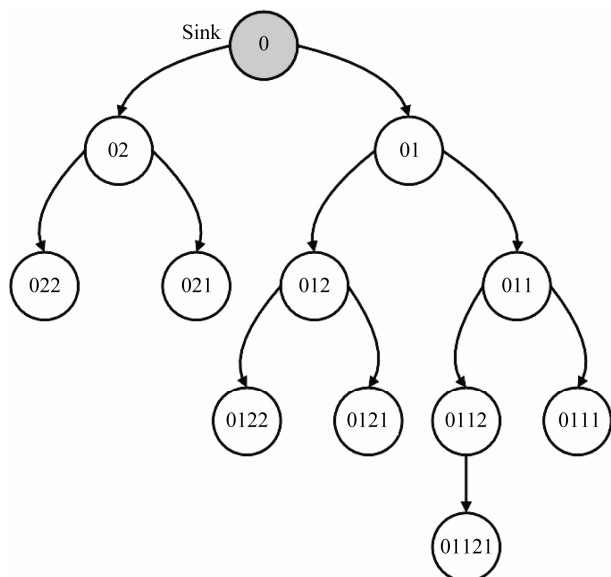
Table 2. Neighbors table structure.

Value	Description
Neighbor ID	This value is contained in the Ready message sent by the neighbor.
Neighbor Power	This value is contained in the Ready message sent by the neighbor.
Neighbor Distance	This value is calculated by the node when it receives the Ready message from the neighbor. The calculation is done according to the strength of the received signals at the physical layer.
Neighbor RAVG	This value is contained in the Ready message sent by the neighbor. It represents the status of the neighbor's neighbors (the characteristics of the nodes around the neighbor).
Neighbor FR	This value is calculated by the node when it receives the Ready message from the neighbor. It is calculated using the Fuzzy Ranking (FR) system (discussed later in this paper). It represents the characteristics of the neighbor itself.
Is Child	This flag is set to 1 if the node sends an Engagement-Acceptance message to the neighbor; therefore the node becomes its parent.
Is Parent	This flag is set to 1 if the node receives an Engagement-Acceptance message from the neighbor; therefore the node becomes its child.

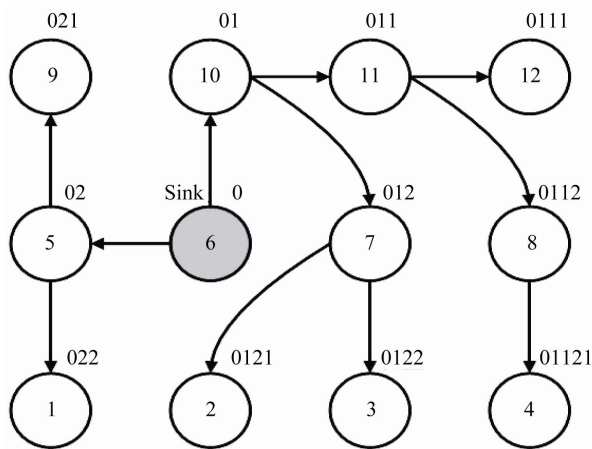
Table 3. Tree construction. Each node will do the following.

Step 1. Wait for Ready message when a one is received then go to step 2.
Step 2. Store message information in the neighbors table and check the ID if it is "null" then go to step 3.
Step 3. Wait for a predefined period of time and store any Ready messages information received during this period, then go to step 4.
Step 4. Calculate the fuzzy rank for each stored neighbor and then go to step 5.
Step 5. Send Engagement message to the neighbor with the best rank. If Engagement-Acceptance message is received, then go to step 6, else go to step 7.
Step 6. Calculate its ID and its rank average then broadcast a Ready message. Then go to step 8.
Step 7. Exclude the best neighbor and go back to step 5.
Step 8. Exit.

an example on logical tree construction. **Figure 1(a)** represents the logical tree view with the new nodes' IDs and **Figure 1(b)** represents the physical distribution of sensor nodes. We assume that node 6 is the sink node and it will initiate the tree construction. The initial ID in this scenario is 0 and C_{\max} (number of children) = 2, thus m (number of digits) = 1. Both nodes 5 and 10 are engaged with the sink which sends them 1 and 2 as Offered



(a)



(b)

Figure 1. Logical tree construction.

IDs. Then they will calculate their ID by concatenating the Offered ID with sink ID. Next, both nodes 5 and 10 send Ready message to their neighbors. This process continues until all nodes successfully engaged with some parent.

3.3.2. Stage 2: New Node Engagement

When a new node wants to join a network, it should broadcast a New Node message. Then all neighbors will reply by either Ready message if the number of children $< C_{max}$, or Unready message if the number of children = C_{max} . The new node will store all information in its neighbors table and calculate the fuzzy rank number for each Ready message. Then it will choose the parent that has the maximum rank, and broadcast a Ready message. The steps are summarized in **Table 4**.

Table 4. New node engagement. New nodes will do the following.

-
- Step 1.** Broadcast New Node message and go to step 2.
 - Step 2.** Wait for Ready message when one is received, then store its information and go to step 3.
If any Unready message is received while waiting then store its information in the neighbors table.
 - Step 3.** Wait for predefined period and store any Ready or Unready message information received during this period then go to step 4.
 - Step 4.** Calculate the fuzzy rank for each Ready message's information that is stored in the neighbor table and go to step 5.
 - Step 5.** Send Engagement to the neighbor with the best rank. If Engagement-Acceptance is received then go to step 6 else go to step 7.
 - Step 6.** Calculate the ID and the rank average then broadcast a Ready message. Then go to step 8.
 - Step 7.** Exclude the best neighbor and go back to step 5.
 - Step 8.** Exit.
-

3.3.3. Stage 3: Message Transmission

As stated earlier the constructed tree should be sink-rooted and all other nodes will send data to it. The message should be forwarded over the best path. To choose the next hop, the sender will consider both neighbors depth and power. The neighbor that has the minimum depth and a power larger than a specific threshold will be chosen. If all small-depth neighbors have critical energy then the sender will send the data to the parent. In this way, the load is balanced among nodes instead of overloading the parent node as in TR [14] or the less depth neighbor node as in [15,21-23]. The fuzzy ranking is not used in this stage since it is not feasible to calculate the rank when data packets need to be sent. The fuzzy ranking is only applied to control packets in order to ensure a balanced topology among nodes.

3.3.4. Stage 4: Tree Reconstruction

If a node's energy reaches a specific threshold, it should inform its parent, children, and neighbors that it will go down, so they can take an action and prepare themselves to reconstruct the tree. Each node has a relation with the dead node should take an action regarding to the relation connecting them. There are three cases; the first one when the dead node is a parent. In this case the children have to find a new parent. Each child broadcasts a Request Parent message and only neighbors with children less than C_{max} will reply by a Ready message, other nodes send Unready messages. The child then calculates the fuzzy rank number for each Ready message and then chooses the node that has the maximum rank as a parent.

Then it will broadcast a Change ID message to its neighbors to update the ID in their neighbors' tables. If any neighbor is a child for this node, it will subsequently change its ID and broadcast a Change ID message. This process continues until all IDs are modified.

The second case is when the dead node is a child. In this case the parent should remove this node from its neighbors table and decrement the number of children. Finally, the last case is when the dead node is a neighbor, then neighbors will remove it from their neighbors' tables.

In some cases nodes go down before informing other nodes that their power is about to end. In this case any neighbor node (could be child or parent) discover the absence of this node, should broadcast the dead node ID in an Inform message and then each node will take an action as discussed above. The steps are illustrated in Table 5 and Figure 2.

3.4. FEAR Analysis

In this section we will analyze FEAR protocol in terms of the number of generated sent and received control messages and the consumed power according to these messages during the tree construction and compare them with PTR protocol [15]. PTR was chosen for comparisons

Table 5. Tree reconstruction. Each node will do the following.

-
- Step 1.** When any Inform message is received then go to step 2.
- Step 2.** Remove the dead node from the neighbors table and check the relation with it. If it is a parent then go to step 3 if it is a child then go to step 4.
- Step 3.** Broadcast Request Parent message and go to step 5.
- Step 4.** Decrement the number of children and go to step 11.
- Step 5.** Wait for Ready message when a one is received then store its information and go to step 6. If any Unready message is received while waiting, then store its information in the neighbors table.
- Step 6.** Wait for a predefined period and store any Ready or Unready message information received during this period then go to step 7.
- Step 7.** Calculate the fuzzy rank for each Ready message's information that is stored in neighbors table and go to step 8.
- Step 8.** Send Engagement to the neighbor with the best rank. If Engagement-Acceptance is received then go to step 9 else go to step 10.
- Step 9.** Calculate the new ID and broadcast Change ID message. Then go to step 11.
- Step 10.** Exclude the best neighbor and go back to step 8.
- Step 11.** Exit.
-

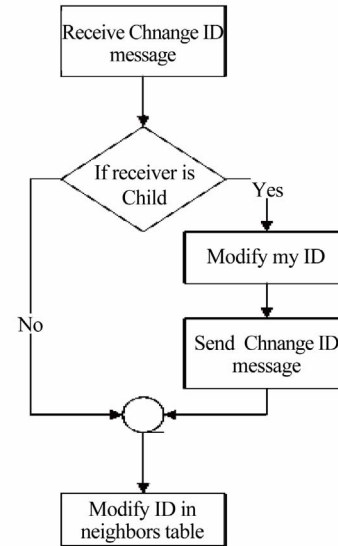


Figure 2. Change ID flowchart.

as it constructs the tree topology with the minimum cost, the same as TR. In addition, PTR provides solutions to recover from node(s)/link(s) similar to FEAR protocol.

3.4.1. Number of Sent and Received Messages

In this subsection we will compare the number of sent and received messages in both FEAR and PTR protocols. For both protocols the worst case has been calculated since the best case is hard to be forecasted.

We derived the number of sent messages in PTR as $3N + 2\sum_{i=1}^N Ne(n_i)$. Each node sends one Association message, for N nodes, there will be N Associations. In the worst case each node will send a Reply message to each Association from each neighbor, if we represent each neighbor set by $Ne(n)$, then that requires the summation of all neighbor sets, $\sum_{i=1}^N Ne(n_i)$. If we

assume that at the end of the tree construction each node will successfully get an ID then this requires sending N messages that contains the IDs. Finally, when the tree is constructed, each node will broadcast its ID in a hello_neighbor message to construct the neighbors table. The total is N hello_neighbor messages. On the other hand, each node will reply by sending a reply_hello_neighbor message to all neighbors and this requires additional $\sum_{i=1}^N Ne(n_i)$ messages. Therefore, a total of $3N + 2\sum_{i=1}^N Ne(n_i)$ control messages are sent in the PTR protocol.

The number of sent messages in FEAR is $2N +$

$\sum_{i=1}^N Ne(n_i)$ as we prove using *Theorem 1*.

Theorem 1: In FEAR protocol, the number of sent control messages is at most $2N + \sum_{i=1}^N Ne(n_i)$

Proof: Three control messages are exchanged between nodes during tree construction; according to *Ready* messages each node will send one *Ready* to broadcast its ID, so for N nodes there will be N messages assuming that all nodes got IDs. For *Engagement* Messages the worst case is to send an engagement to each neighbor, so for N nodes if each node has a set of neighbors represented by $Ne(n)$, the total is the summation of all neighbor sets which is equal to $\sum_{i=1}^N Ne(n_i)$. Finally, assuming

that each node will be successfully engaged to one parent and gets an *ID* there should be N Engagement-Acceptance messages. Therefore, a total of $2N + \sum_{i=1}^N Ne(n_i)$ control messages are sent in the FEAR protocol.

We derived the number of received messages in PTR as $N + 4\sum_{i=1}^N Ne(n_i)$ messages. Each node receives an Association from all neighbors in its neighbors set, the total is the summation of all neighbor sets $\sum_{i=1}^N Ne(n_i)$.

On the other hand, each node in the worst case expects to receive a Reply to its Association from all of its neighbors and that requires another $\sum_{i=1}^N Ne(n_i)$ messages.

Finally, if each node successfully associates to one parent, then it will receive an ID and that requires N ID messages to be received. For constructing the neighbors table each node will collect *hello_neighbor* messages and *reply_hello_neighbor* messages from its neighbors and that needs to receive $\sum_{i=1}^N Ne(n_i)$ messages for each

message type. Therefore, a total of $N + 4\sum_{i=1}^N Ne(n_i)$ is the number of received control messages in the PTR protocol. The number of received messages in FEAR protocol is $N + 2\sum_{i=1}^N Ne(n_i)$ messages as we prove using *Theorem 2*.

Theorem 2: In FEAR protocol, the number of received control messages is at most $N + 2\sum_{i=1}^N Ne(n_i)$

Proof: Three control messages are used in tree construction. Firstly, each node broadcast its ID to its

neighbors in a *Ready* message, if we represent each neighbor set by $Ne(n)$ the total is the summation of all Sets, $(\sum_{i=1}^N Ne(n_i))$. Secondly, for *Engagement* messages

the worst case occurs when each node will receive engagements from all neighbors in its set. So the total is the summation of all neighbor sets, i.e., $\sum_{i=1}^N Ne(n_i)$.

Finally, each node will be engaged to only one parent and will receive only one *Engagement-Acceptance* message, so the total is N messages. Therefore, a total of

$N + 2\sum_{i=1}^N Ne(n_i)$ is the number of received control messages in the FEAR protocol.

3.4.2. Consumed Power

Sensor power is affected by local processing and communication operations. Since communication operations consume more power than data processing, sensors will lose most of its power during sending and receiving of messages [2]. According to [24] a node requires $E_{Tx}(k, d)$ to send k bits message to destination at distance d , and $E_{Rx}(k)$ to receive k bits message. $E_{Tx}(k, d)$ and $E_{Rx}(k)$ are defined as:

$$\begin{aligned} E_{Tx}(k, d) &= E_{Tx-elec}(k) + E_{Tx-amp}(k, d) \\ &= E_{elec} * k + \epsilon_{amp} * k * d^2 \end{aligned} \quad (1)$$

$$\begin{aligned} E_{Rx}(k) &= E_{Rx-elec}(k) \\ &= E_{elec} * k \end{aligned} \quad (2)$$

where $E_{elec} = 50$ nJ/bit and $\epsilon_{amp} = 100$ pJ/bit/m².

Using (1) and (2), the maximum power that will be consumed during the tree construction can be calculated according to *Theorem 1* and *Theorem 2*. The power that is consumed by sending and receiving control messages during the FEAR tree construction phase can be calculated using (3).

$$\begin{aligned} \text{FEAR Consumed Power} &= E_{Tx}(k, d) * \left(2N + \sum_{i=1}^N Ne(n_i) \right) \\ &+ E_{Rx}(k) * \left(N + 2\sum_{i=1}^N Ne(n_i) \right) \end{aligned} \quad (3)$$

4. Fuzzy Ranking System

In the FEAR protocol, we have built a fuzzy inference system that will be used in tree construction, tree reconstruction, and new node engagement phases. The purpose of this system is to assign a rank for each neighboring node. This ranking helps the node to be associated with the best possible neighbor, so the tree is balanced ac-

cording to both nodes energy and depth. The general structure for the fuzzy ranking system is shown in **Figure 3**. This ranking system consists of three stages. The output from each stage is one of the inputs to the next stage. The fuzzy inference that is used in all stages is based on Mamdani fuzzy inference [25].

4.1. Neighbors Classification

In order to rank neighbors, they are classified into four types. Each neighbor node should belong to only one type in a particular point of time. **Table 6** illustrates these types. This classification is used to know how good or bad the neighbors are. Neighbors belong to the first type will take higher rank than other types, so the larger the number of neighbors belong to this type, the better the performance of our protocol since many good neighbors can be utilized as intermediate nodes instead of parent node.

4.2. First Stage of Fuzzy Ranking System

This stage has two-input, one-output fuzzy inference system. It finds the cost of transmitting the data to a particular neighbor in terms of neighbor’s depth and distance from the sender. The output from this stage is entered to the next stage. **Figure 4** shows the structure of this stage. The inputs are mapped to fuzzy membership function illustrated in **Figure 4 (a)**. The first input is the distance which can be Very_Near, Near, Far, and Very

Far. Usually the transmission range for sensor devices is about 250 meters, so any neighboring node should be placed within this range. This input can be calculated using signal strength [9]. The second input to this stage is the depth which can be calculated from the neighbor ID that is stored in the neighbors table. The depth is mapped into Small, Medium, and Large membership functions and ranges from 1 to maximum_ tree_depth. The maximum_tree_depth is calculated according to (4).

$$\text{Maximum_Tree_Depth} = \log_{C_{\max}} N \quad (4)$$

where:

- N is the number of network nodes, and
- C_{\max} is the maximum number of children.

Both distance and depth affect the transmission cost; larger depth implies larger number of intermediate nodes. If two neighbors have the same depth then it is more feasible to forward the message to the nearest one since short distance requires less power to send the signal. Transmission_cost is mapped into Low, Medium, and High membership functions as illustrated in **Figure 4(b)**.

4.3. Second Stage of Fuzzy Ranking System

The second stage inputs are Transmission_cost with Low, Medium, and High membership functions and the neighbor residual_energy that could be Low, Medium, and High as illustrated in **Figure 5(a)**. The residual_energy for each neighbor is stored in the neighbor’s table. The output from this fuzzy stage is the neighbor_rank.

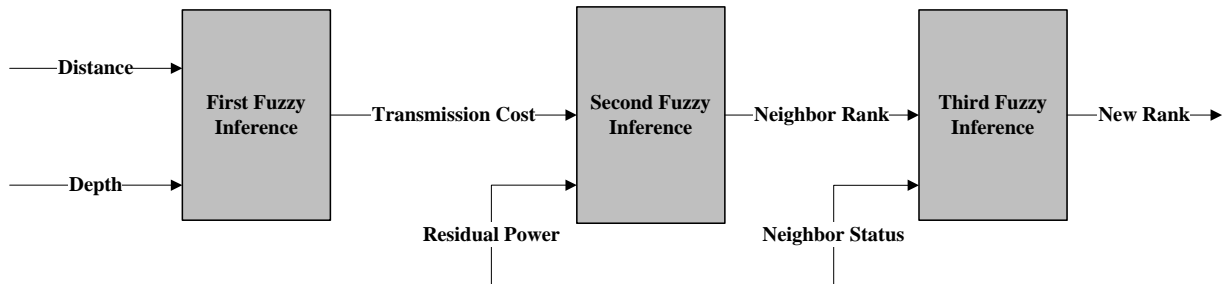
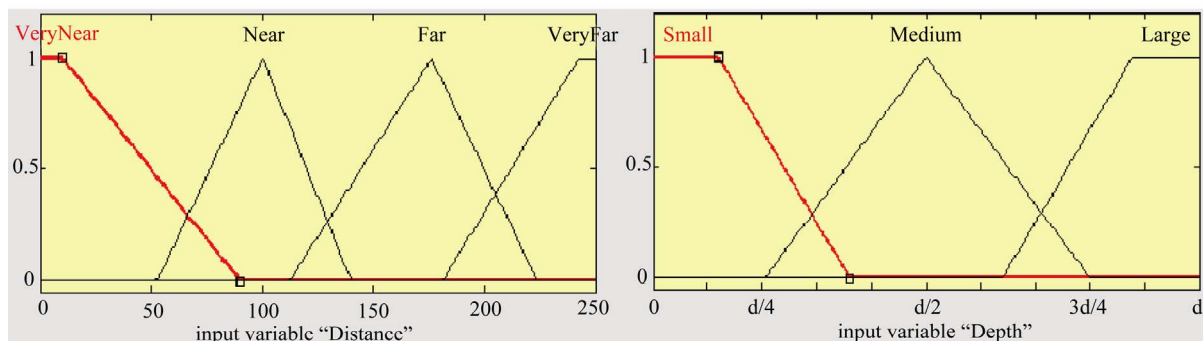


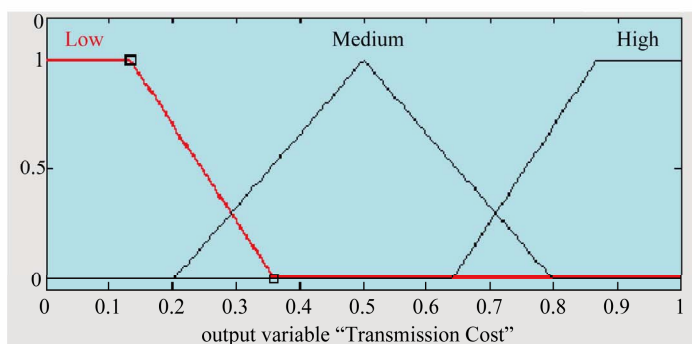
Figure 3. Fuzzy ranking system.

Table 6. Neighbors types.

Type	Description
Good depth and Good energy	The node belongs to this type has a residual energy greater than a specific threshold, and a depth not larger than sender’s parent depth.
Good depth and Bad energy	The node belongs to this type has a small amount of residual energy, and a depth not larger than sender’s parent depth.
Bad depth and Good energy	The node belongs to this type has a residual energy greater than specific threshold, but its depth is larger than sender’s parent depth.
Bad depth and Bad energy	The node belongs to this type has a small amount of residual energy, and its depth is larger than sender parent depth.

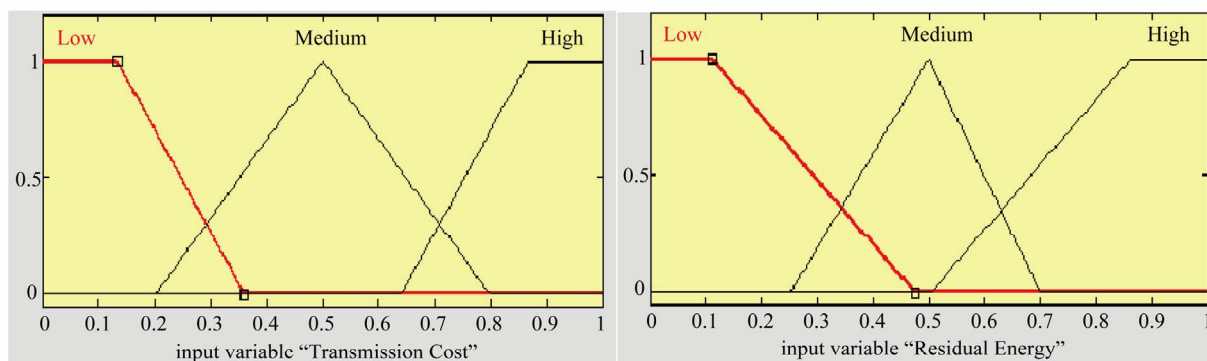


(a)

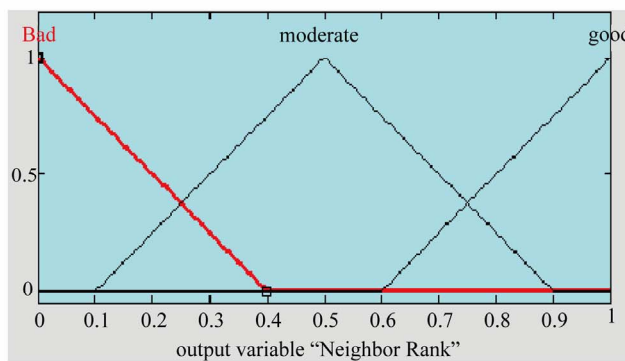


(b)

Figure 4. First stage fuzzy. (a) The inputs membership functions; (b) The output membership function.



(a)



(b)

Figure 5. Second stage fuzzy. (a) The inputs membership functions; (b) The output membership function.

This rank is mapped into Good, Moderate, and Bad. See **Figure 5(b)**.

4.4. Third Stage of Fuzzy Ranking System

This final stage is used to find the final neighbors rank according to their characteristics and status. It takes two inputs; the neighbor_rank that resulted from the previous stage and the neighbor_status. The neighbor_rank gives an indication about the neighbor characteristic in terms of residual_power, depth and distance. These factors will be used to rank the neighbor; the larger the rank, the better the neighbor chance to be chosen as a parent. **Figure 6** shows the structure for this fuzzy stage. The neighbor_status is mapped into Good, Moderate, and Bad fuzzy membership functions. This input gives an indication about the characteristics of the nodes around the neighbor (neighbors of a neighbor) and is calculated by the neighbors themselves as the following:

- When a node receives a New Node, Request Parent, or Engagement-Acceptance message, it will calculate the rank for each neighbor using fuzzy ranking system.
 - Calculate the average of neighbor ranks. Since power and depth are balanced among neighbors, average will be a good measure to reflect the status of the neighbors.
 - Send a Ready message that contains the average value.
- Now each node receives the Ready message will use the average value as the second input to the third stage

inference. The higher the average value, the better the neighbor’s status and the better the chance to be chosen as a parent. We assume that the Ready message that is sent by the sink will have the highest possible rank average since it is the best choice for any neighbor to be associated with.

5. Simulation Results and Comparison

This section discusses the results obtained using FEAR and compare it with TR [14] and PTR [15] protocols. We use Java programming language to build our simulator. Different evaluation metrics are considered in our simulation that will be discussed in the following subsections. **Table 7** shows the simulation parameters that were used.

5.1. Network Overhead

The network overhead is evaluated in terms of number of messages that are sent and received during tree construction. We use different network sizes 25, 50, 100 and 500. For each size, the same nodes characteristics (initial power, nodes distribution, and distance from sink) are used in the three protocols. We take the average of 10 simulation runs. **Figures 7** and **Figure 8** show the behavior of the three protocols according to the number of sent and received messages, respectively. It can be noticed that TR curve does not appear as it is almost identical to FEAR curve, therefore, the results are also illus-

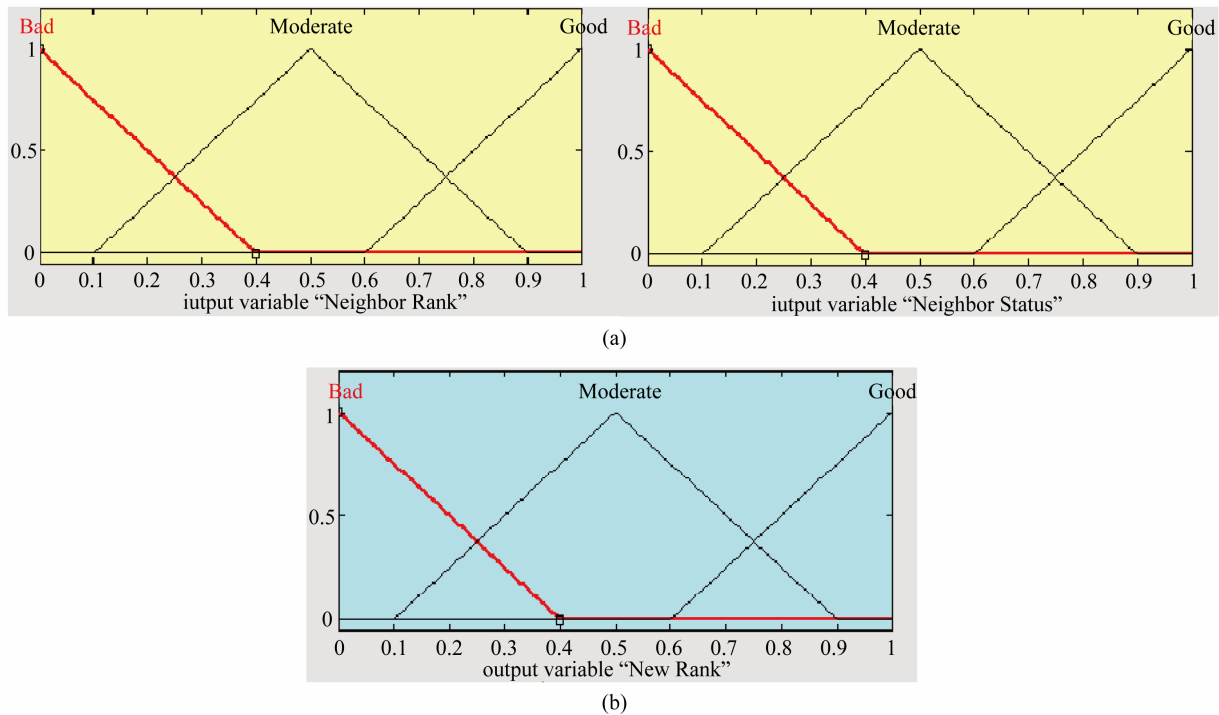
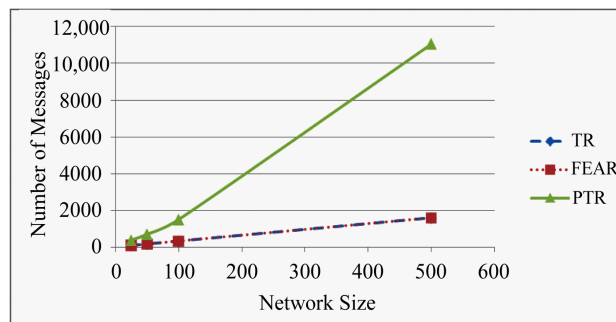
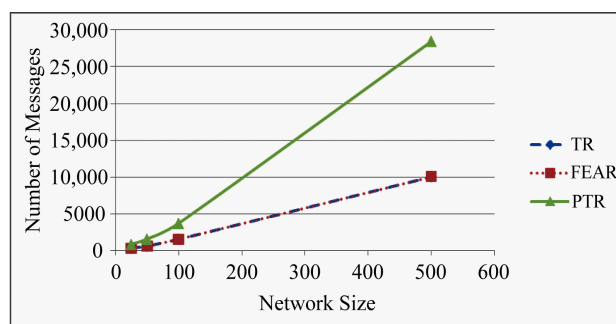


Figure 6. Third stage fuzzy. (a) The inputs membership functions; (b) The output membership function.

Table 7. Simulation parameters. Area corresponding to each network size.

Simulation parameter	Value
Network Size	25, 50, 100, 500
Terrain Area (m ²)	500 × 600, 800 × 1000, 1000 × 1250, 2000 × 2500
Mobility	None
Radio Range	250 m

**Figure 7. Number of sent messages with different network sizes.****Figure 8. The overhead of received messages with different network sizes.**

trated in **Table 8** for more precise comparison. As illustrated earlier, the proposed FEAR protocol has the same behavior as TR protocol since it uses the minimum possible messages to construct the tree, it also utilizes these messages to construct the neighbor tables without the need to send additional Hello and Hello Reply messages as used in PTR protocol to aid in node(s)/link(s) failure recovery. According to the results, FEAR protocol saves up to 85.67% in the number of sent messages and up to 64.6% in the number of received messages. Overall, FEAR protocol saves up to 70.5% in the number of control messages.

5.2. Network Power Consumption

As stated in the analysis section (Section 3.4) most of the network's power is consumed by the communication operations. Therefore, we consider the number of sent and received messages in the energy consumption evaluation.

Table 8. Network overhead.

	Network Size	FEAR	TR	PTR
Sent Messages	25	87	89	362
	50	164	168	689
	100	328	329	1488
	500	1585	1590	11064
Received Messages	25	313	315	832
	50	586	590	1561
	100	1521	1528	3697
	500	10068	10072	28444

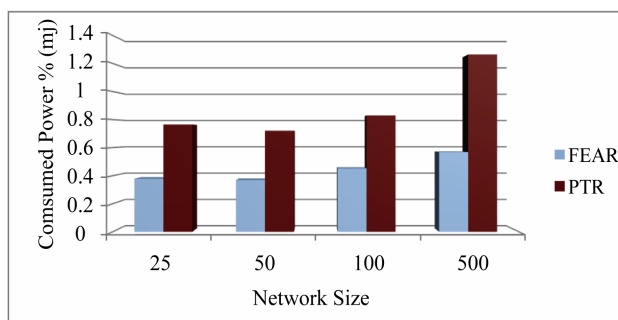
lation. The consumed power is compared with PTR protocol since it constructs both tree and neighbors' tables as FEAR protocol, whereas TR constructs only the tree. We use the same simulation characteristics that are used in the overhead evaluation. We take the average of 10 runs for each network size and compute the consumed power according to (3) (we ignored the distance in the comparison since we use the same nodes distribution for both protocols). The results are shown in **Table 9** and **Figure 9**. As illustrated, FEAR protocol is better than PTR protocol since it reduces the control messages that should be exchanged between nodes. It decreases the consumed power by up to 55.08% comparing with PTR protocol.

5.3. Transmission Delay

Transmission delay is evaluated according to the number of intermediate nodes between the sender and the sink. Transmission delay is affected by sender's depth; the deeper the node, the larger the number of intermediate nodes. The same constructed tree is used for the three considered protocols and the average of different sce-

Table 9. Network consumed power according to network overhead.

	Network Size	FEAR	PTR	FEAR/PTR (%)
Consumed Power (mj)	25	0.9462237	1.911558	0.4950013
	50	1.8420064	3.602205	0.5113552
	100	4.5128878	8.300762	0.5436715
	500	28.4113085	63.2482	0.4492034

**Figure 9. The percentage of the consumed power with different network sizes.**

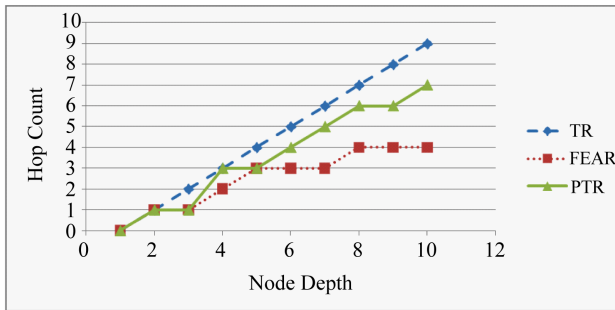


Figure 10. Hop count according to different nodes depth.

Table 10. Failure scenarios. Success: no isolation, Fail: isolation.

Scenario	TR	PTR	FEAR
Dead node is a leaf node	Success	Success	Success
Dead node is a parent and all of its children are in sink range or each child has at least one descendent in sink range	Fail	Success	Success
Dead node is a parent but all or some of its children and descendents are not in sink range	Fail	Fail	Success
Dead node is a parent but all or some of its children do not have neighbor(s). (Become isolated after parent death).	Fail	Fail	Fail

narios is computed. Figure 10 illustrates the results. As shown in the figure, FEAR protocol has the minimum delay comparing to the TR and PTR protocols.

5.4. Possibility to Recover from Failure

This subsection analyzes different failure scenarios. For each scenario, the possibility to reconstruct the tree and to recover from the failure is compared with the TR and PTR protocols. The same network characteristics are used for the three protocols. We assess whether the scenario is successful or not by performing many simulation runs for each scenario on different network sizes. As illustrated in Table 10, TR protocol does not provide failure recovery mechanisms whereas both FEAR and PTR can recover from failure. The results show that FEAR solutions are more efficient in reconstructing the tree than the one provided by PTR.

6. Conclusions and Future Work

In this paper we propose a tree-based routing protocol for Wireless Sensor Networks (WSNs) that considers both shortest path and energy balance between nodes. The proposed protocol is called Fuzzy-based Energy Aware Routing Protocol (FEAR) as it uses a fuzzy inference

system to rank the nodes. This is used to ensure that each node is associated with the best neighbor in the tree construction. The protocol provides an energy efficient solution for both data routing and network failure, thus prolongs the network’s life time. FEAR has several stages: sink-rooted tree construction, messages transmission, and node/link failure problem recovery.

The protocol is implemented and compared to other tree-based protocols. The simulation results show that the new protocol saves up to 70.5% in the number of control messages and up to 55.08% in the consumed power comparing with other related work. As a future work we will consider security issues and possible threats to develop a secure FEAR protocol.

7. References

- [1] R. Biradar, V. Patil, S. Sawant and R. Mudholkar, “Classification and Comparison of Routing Protocols in Wireless Sensor Networks,” *Special Issue on Ubiquitous Computing Security Systems, UbiCC Journal*, Vol. 4, 2009, pp. 704-711.
- [2] A. Jamalipour and J. Zheng, “Wireless Sensor Networks: A Networking Perspective,” Wiley-IEEE Press, Hoboken, 2009.
- [3] M. McGrath and T. Dishongh, “Wireless Sensor Networks for Healthcare Applications,” Artech House, London, 2010.
- [4] P. Pandian, “Wireless Sensor Network for Wearable Physiological Monitoring,” *Journal of Networks*, Vol. 3, No. 5, 2008, pp. 21-29. doi:10.4304/jnw.3.5.21-29
- [5] H. Alemdar and C. Ersoy, “Wireless Sensor Networks for Healthcare: A Survey,” *Computer Networks*, Vol. 54, No. 15, 2010, pp. 2688-2710. doi:10.1016/j.comnet.2010.05.003
- [6] S. Diamond and M. Ceruti, “Application of Wireless Sensor Network to Military Information Integration,” *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, Vol. 1, 2007, pp. 317-322. doi:10.1109/INDIN.2007.4384776
- [7] M. Winkler, K. Tuchs, K. Hughes and G. Barclay, “Theoretical and Practical Aspects of Military Wireless Sensor Networks,” *Journal of Telecommunications and Information Technology*, Vol. 2, 2008, pp. 37-45.
- [8] X. Shen, Z. Wang, and Y. Sun, “Wireless Sensor Networks for Industrial Applications,” *5th World Congress on Intelligent Control and Automation*, Vol. 4, 2004, pp. 3636-3640. doi:10.1109/WCICA.2004.1343273
- [9] J. Al-Karaki and A. Kamal, “Routing Techniques in Wireless Sensor Networks: A Survey,” *IEEE Wireless Communications*, Vol. 11, No. 6, 2004, pp. 6-28. doi:10.1109/MWC.2004.1368893
- [10] L. Villalba, A. Orozco, A. Cabrera and C. Abbas, “Routing Protocols in Wireless Sensor Networks,” *Sensors*, Vol. 9, No. 11, 2009, pp. 8399-8421. doi:10.3390/s91108399

- [11] L. Wang, C. Wang and C. Liu, "Optimal Number of Clusters in Dense Wireless Sensor Networks: A Cross-Layer Approach," *IEEE Transactions on Vehicular Technology*, Vol. 58, No. 2, 2009, pp. 966-976. [doi:10.1109/TVT.2008.928637](https://doi.org/10.1109/TVT.2008.928637)
- [12] D. Djenouri and I. Balasingham, "New QoS and Geographical Routing in Wireless Biomedical Sensor Networks," *6th International Conference on Broadband Communications, Networks and Systems*, Madrid, 14-16 September 2009, pp. 1-8. [doi:10.4108/ICST.BROADNETS2009.7188](https://doi.org/10.4108/ICST.BROADNETS2009.7188)
- [13] E. Stavroua and A. Pitsillides, "A Survey on Secure Multipath Routing Protocols in WSNs," *Computer Networks*, Vol. 54, No. 13, 2010, pp. 2215-2238. [doi:10.1016/j.comnet.2010.02.015](https://doi.org/10.1016/j.comnet.2010.02.015)
- [14] IEEE, "ZigBee Specification Version 1.0," ZigBee Alliance, San Ramon, 2005.
- [15] Y. Park and E. Jung, "Plus-Tree: A Routing Protocol for Wireless Sensor Networks," *Lecture Notes in Computer Science*, Vol. 4413, 2007, pp. 638-646. [doi:10.1007/978-3-540-77368-9_62](https://doi.org/10.1007/978-3-540-77368-9_62)
- [16] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," *Ad Hoc Networks*, Vol. 3, No. 3, 2005, pp. 325-349. [doi:10.1016/j.adhoc.2003.09.010](https://doi.org/10.1016/j.adhoc.2003.09.010)
- [17] L. Almazaydeh, E. Abdelfattah, M. Al-Bzoor and A. Al-Rahayfeh, "Performance Evaluation Of Routing Protocols in Wireless Sensor Networks," *International Journal of Computer Science and Information Technology*, Vol. 2, No. 2, 2010, pp. 64-73. [doi:10.5121/ijcsit.2010.2206](https://doi.org/10.5121/ijcsit.2010.2206)
- [18] P. Loh and Y. Pan, "An Energy-Aware Clustering Approach for Wireless Sensor Network," *International Journal of Communications, Network and System Sciences*, Vol. 2, No. 2, 2009, pp. 131-141. [doi:10.4236/ijcns.2009.22015](https://doi.org/10.4236/ijcns.2009.22015)
- [19] O. Zytoune, Y. Fakhri and D. Aboutajdine, "A Novel Energy Aware Clustering Technique for Routing in Wireless Sensor Networks," *Wireless Sensor Network*, Vol. 2, No. 3, 2010, pp. 233-238. [doi:10.4236/wsn.2010.23031](https://doi.org/10.4236/wsn.2010.23031)
- [20] A. Mohajerzadeh and M. Yaghmaee, "Tree Based Energy and Congestion Aware Routing Protocol for Wireless Sensor Networks," *Wireless Sensor Network*, Vol. 2, No. 2, 2010, pp. 161-167. [doi:10.4236/wsn.2010.22021](https://doi.org/10.4236/wsn.2010.22021)
- [21] M. Al-Harbawi, M. Rasid and N. Noordin, "Improved Tree Routing (ImpTR) Protocol for ZigBee Network," *International Journal of Computer Science and Network Security*, Vol. 9, No. 10, 2009, pp. 146-152.
- [22] W. Qiu, E. Skafidas and P. Hao, "Enhanced Tree Routing for Wireless Sensor Networks," *Ad Hoc Networks*, Vol. 7, No. 3, 2009, pp. 638-650. [doi:10.1016/j.adhoc.2008.07.006](https://doi.org/10.1016/j.adhoc.2008.07.006)
- [23] M. Zeynali, L. Khanli and A. Mollanejad, "TBRP: Novel Tree Based Routing Protocol in Wireless Sensor Network," *International Journal of Grid and Distributed Computing*, Vol. 2, No. 4, 2009, pp.35-48.
- [24] W. Heinzelman, A. Sinha, A. Wang and A. Chandrakasan, "Energy-Scalable Algorithms and Protocols for Wireless Microsensor Networks," *International Conference on Acoustics, Speech, and Signal Processing*, Vol. 6, 2000, pp. 3722-3725. [doi:10.1109/ICASSP.2000.860211](https://doi.org/10.1109/ICASSP.2000.860211)
- [25] E. Mamdani, "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, Vol. C-26, No. 12, 1977, pp. 1182-1191. [doi:10.1109/TC.1977.1674779](https://doi.org/10.1109/TC.1977.1674779)