A Segment-based Tree Traversal Algorithm for Enhancing Data Gathering in Wireless Sensor Networks

MOHAMMAD A. JASSIM
Department of Computer Science, KASIT
The University of Jordan Amman,
JORDAN

WESAM A. ALMOBAIDEEN
Department of Computer Science, KASIT
The University of Jordan Amman,
JORDAN

Abstract—Wireless Sensor Networks (WSNs) are sink-based networks in which assigned sinks gather all data sensed by lightweight devices that are deployed in natural areas. The sensor devices are energy-scarce, therefore, energy-efficient protocols need to be designed for this kind of technology. Power-Efficient GAthering in Sensor Information Systems (PEGASIS) protocol is an energy-efficient data gathering protocol in which a chain is constructed using a greedy approach. This greedy approach has appeared to have unbalanced distances among the nodes which result in unfair energy consumption. Tree traversal algorithms have been used to improve the constructed chain to distribute the energy consumption fairly. In this research, however, a new segmentbased tree traversal approach is introduced to further improve the constructed chain. Our new proposed algorithm first constructs initial segments based on a list of nodes that are sorted according to post-order traversal. Afterwards, it groups these segments and concatenates them one by one according to their location; thus, our proposed approach uses location-awareness to construct a single balanced chain in order to use it for the data gathering process. This approach has been evaluated under various numbers of sensor devices in the network field with respect to various crucial performance metrics. It is shown in our conducted simulation results that our proposed segment-based chain construction approach produces shorter chains and shorter transmission ranges which as a result has improved the overall energy consumption per round, network lifetime, and end-to-end delay.

Keywords-component; Wireless Sensor Networks (WSNs); data aggregation; energy-efficient; greedy algorithm; PEGASIS protocol; tree traversal; Minimum Spanning Tree (MST); Segment-Based Tree Traversal (SBTT); cross elimination method; Highest Transmission Range (HTR); C++; QualNet; Time Devision Multiple Access (TDMA); Energy Lost per Round (ELR); network lifetime; end-to-end delay

Received: March 31, 2021. Revised: April 20, 2021. Accepted: April 22, 2021. Published: April 27, 2021.

1. Introduction

Wireless Sensor Networks (WSNs) are sink-based networks in which assigned sinks gather all data sensed by lightweight devices that are distributed and deployed in natural areas [3][9]. They are used for specific applications to monitor certain conditions such as temperature, pressure, physical motions, and humidity. As a result, the special capabilities of WSNs have led to implement it in many areas like environmental monitoring, traffic control and marine applications [2][4][13].

WSNs are also used in highly critical fields such as military, medical health, coal mines, and even nuclear and radiation monitoring where hazardous environments exist [1][9]. Implementing WSNs in such a wide range of critical applications requires a highly efficient design of this technology. Moreover, specific performance metrics are demanded to be considered with the designation process [2][13].

According to [4], the most critical resource in WSNs is energy. Thus, almost every protocol designed with respect to WSNs is as close as possible to the term 'energy-efficient'. Therefore, it is reasonable to focus on energy while designing models in WSNs such that they consume as less energy as possible [9].

Many techniques and mechanisms have been developed to reduce energy consumption in WSNs. Some important techniques are reducing communication distances among the sensor devices and data aggregation [2][8][11].

First, sending and receiving data through long communication ranges and distances consume large amounts of energy in a sensor device [8]. Secondly, the idea of data aggregation is to combine data gathered from various sources in order to eliminate redundancy and minimize the number of transmissions which in turn reduces the overall consumed energy within the network [1][2][11].

2. Literature Review

Many protocols have been developed to minimize the long ranged data transmissions in addition to data reduction through data aggregation technique. Some of the most popular protocols regarding this subject are Low Energy Adaptive Clustering Hierarchy (LEACH) and Power-Efficient GAthering in Sensor Information Systems (PEGASIS) [3][12][13].

PEGASIS is a near optimal chain based protocol. The key idea of PEGASIS is to form a long single chain that connects all nodes in the network which afterwards enables these nodes to communicate and exchange data among each other. The chain that is constructed in PEGASIS is formed using greedy algorithm. In this chain, each node transmits data to a close neighbor. Clearly, this is a simple approach that motivates sensor devices to transmit through shorter distances, thus, reducing energy consumption. Additionally, while nodes are transmitting data, they also aggregate and fuse data along the way [12].

Using greedy algorithms for chain formation in PEGASIS protocol rarely yields optimal solutions. Therefore, it is hypothesized that node lifetime and energy consumption in PEGASIS protocol whose chain is built using greedy approach can be further improved [10].

The work in [10] is mainly aimed at using the traditional graph theory tree traversal algorithms to form the chain in PEGASIS instead of using the greedy algorithm in [12]. First, a rooted directed minimum-weight spanning tree of the whole WSN is generated. To construct the rooted directed minimum spanning tree, Kruskal's algorithm is used to obtain an undirected unrooted minimum spanning tree. Then, a Breadth First Search (BFS) algorithm is run on the tree to obtain a rooted directed minimum spanning tree. Furthermore, it is important to mention that the root of the tree is the node that is closest to the center of the network area. At the end, post-order tree traversal algorithm is used to form a sequence of ordered nodes based on the rooted directed MST; this chain is used in PEGASIS data gathering protocol [10].

Simulations have been conducted in [10] for three types of tree traversal algorithms (pre-order, in-order and post-order); results show that post-order traversal outperforms other tree traversal methods, thus, we will only consider this traversing algorithm in our research.

It is reported in [10] that the post-order approach has a longer network lifetime than the greedy approach in [12] for TDMA systems. This is explained in [10] by referring to the fact that the physical distances among the nodes in the greedy chain get larger when more nodes are added to the chain. Furthermore, the distances among the nodes in the second half of the chain are larger than the distances in the first half of the chain. Due to these reasons, premature node failures occur more often in the second half of the chain compared to the first half. In tree traversal approach, on the other hand, the nodes are relatively more equally spaced; this results in less consumption in the overall energy and it does not vary much along the chain. Another important point mentioned in [10] is that choosing the tree's root node to be closest node to the center of the network field yields a tree with greater depth; hence, node lifetime has significantly improved. However, it is concluded in [10] that the greedy approach records better energy lost per round results than the post-order approach.

3. The Proposed Idea

Transmitting data over a wireless medium consumes a relatively big amount of energy. It is important to note that transmitting one bit of data consumes more energy than processing this data in a sensor device. In other words, energy consumption resulting from communication is higher than energy consumption resulting from computation [8]. Thus, energy consumption can be reduced by reducing the communication range; as a result, decreasing the distances among the communicating sensor devices can expand the network lifetime.

We propose in this research paper a chain formation approach that aims to produce a more balanced and efficient chain than the one suggested in [10]; this chain is used for the data gathering process in PEGASIS protocol [10][12]. We believe that producing an improved chain can result in an improved data gathering process, thus, improving the network's overall performance. Our goal is to improve certain performance metrics such as energy lost per round [10], network lifetime [10] and end-to-end delay [14][15]. We define our proposed algorithm as Segment-Based Tree Traversal algorithm (SBTT).

In reality, our actual work starts after obtaining a chain of post-order traversal using the same methods proposed in [10]. SBTT algorithm is consisted of two phases. The first phase is concerned with segments construction, while the second phase groups directly connected segments and incrementally concatenate them one by one until a single segment is conducted.

3.1 Formal Algorithm

The first phase of SBTT algorithm is illustrated in Algorithm 1. Note that *N* represents the number of nodes in the network field. The algorithm initially starts with *POS* which is a list of *N* vertices (nodes) that are ordered according to post-order traversal. Another list known as *list_of_segments* contains the existing segments. Therefore, *list_of_segments* is a list of lists and initially it contains only one segment which is $segment_1$. In addition, $segment_1$ is initially empty (it contains no nodes initially). Moreover, two variables *i* and *j* are defined and initially assigned the value 1. Variable *i* is used to move incrementally along the nodes in *POS* while *j* is used to move along the segments in *list_of_segments*. Furthermore, the two conditions in the **if** statement in lines 8 and 9 are processed with respect to the rooted directed minimum spanning tree.

Algorithm 1 First phase of SBTT algorithm

Input: Post-order List (*POS*) and the Rooted-Directed Minimum Spanning Tree (*directed-MST (V, E)*)

Output: List of Segments (*list_of_segments*)

1: *POS* := { $V_1, V_2, V_3, \dots, V_N$ }
2: *list_of_segments* := { $segment_1$ }
3: $segment_1$:= { Φ }
4: $i \leftarrow 1$
5: $j \leftarrow 1$
6: **loop$_1$ until** $i > N$
7: add V_i to end of $segment_j$
8: **if** V_i in *POS* has **more than** one child **or** V_{i+1} **is not**
9: parent of V_i **then**
10: create a new empty $segment_{j+1}$
11: add $segment_{j+1}$ to rear of *list_of_segments*
12: increment *j*
13: increment *i*
14: go back to line 6
15: **end-if**
16: increment *i*
17: **end-loop$_1$**

After constructing the required segments in phase 1, we move to phase 2 which is illustrated in Algorithm 2. Initially, all segments in *list_of_segments* are marked. The purpose of marking and unmarking segments is to prevent segments that have been concatenated from being processed and concatenated again in the same iteration. Moreover, the value α represents the total number of segments in *list_of_segments*. Additionally, the first condition in the **if** statement checks if there are two nodes in each of the currently processed segments are directly connected with respect to the minimum spanning tree. This specific condition is the key step to connect one segment to another one that is close to it (if there exists a short link between its nodes) which minimizes the long transmission ranges in post-order chain.

To illustrate the process in lines 11-14, suppose that we intend to concatenate the two segments { *a, b, c* } and { *d, e, f* }. According to the algorithm, the outcome concatenated segment should look like this: { *d, e, f, a, b, c* } since the first segment is added to the end of the second segment. However, to improve the distances among the nodes in the segment, a reversing technique is used to connect the closest ends of each segment. The ends of the first segment are nodes *a* and *c* while the ends of the second segment are nodes *d* and *f*. Consider the case in which the distance between nodes *c* and *d* is shorter than the distance between nodes *a* and *f* (which are connected in the concatenated segment { *d, e, f, a, b, c* }). Therefore, a modification needs to take place in order to concatenate the two segments in a way that nodes *c* and *d* become the connecting points of the two segments instead of nodes *a* and *f*. As a result, a reversing operation on both segments before connecting them yields the following concatenated segment: { *f, e, d, c, b, a* }. Hence, a shorter bridge between the two segments is constructed by reversing one of the segments (or both) before concatenating them. These procedures are repeated until all the segments that have been created initially are fused into one single segment. After achieving this one single segment in line 24, Cross Elimination method is applied on this segment. The process of this method takes place in lines 25-31 in Algorithm 2.

Algorithm 2 Second phase of SBTT algorithm

Input: List of Segments (*list_of_segments*)

Output: Segment-based Chain

1: unmark all segments in *list_of_segments*
2: $j \leftarrow 1$
3: $\alpha \leftarrow$ number of segments in *list_of_segments*
4: **loop$_1$ until** *list_of_segments* contains only one segment
5: **loop$_2$ until** $j > \alpha$
6: $k \leftarrow 1$
7: **loop$_3$ until** $k > \alpha$
8: **if** $segment_j$ and $segment_k$ are directly
9: connected in *MST (V, E)* **and** $segment_j$
10: and $segment_k$ are unmarked **and** $j \neq k$ **then**
11: add $segment_j$ to rear of $segment_k$ with
12: reversing segments if necessary such that
13: the closest ends of each segment are
14: connected
15: remove $segment_j$ from *list_of_segments*
16: mark $segment_k$
17: go back to line 5
18: **end-if**
19: increment *k*
20: **end-loop$_3$**
21: increment *j*
22: **end-loop$_2$**
23: unmark all segments in *list_of_segments*
24: **end-loop$_1$**
25: *front_node* ← 1
26: *rear_node* ← 5
27: **loop$_4$ until** *rear_node* > *N*
28: apply Cross Elimination method on nodes from
29: *front_node* to *rear_node* in Segment-based Chain
30: increment *front_node* and *rear_node*
31: **end-loop$_4$**

Cross Elimination method [6] is used to further improve the distances among the nodes in a chain. This mechanism adjusts the path among four selected nodes. However, we extend this concept into selecting five nodes. For example, suppose that we intend to adjust the path along the five nodes: *1, 2, 3, 4, 5*. Hence, we examine all the possible combinations that represent distinct paths from node 1 to node 5. At the end, the path that yields the shortest distance from node 1 all the way to node 5 is selected and according to which, the path from node 1 to node 5 is modified and adjusted. Therefore, the following distances are calculated, and the shortest version is chosen and based on which the path is adjusted: (where *d (x, y)* is the physical distance between node *x* and node *y*)

- *d (1, 5) = d (1, 2) + d (2, 3) + d (3, 4) + d (4, 5)*
- *d (1, 5) = d (1, 3) + d (3, 2) + d (2, 4) + d (4, 5)*
- *d (1, 5) = d (1, 3) + d (3, 4) + d (4, 2) + d (2, 5)*
- *d (1, 5) = d (1, 4) + d (4, 3) + d (3, 2) + d (2, 5)*
- *d (1, 5) = d (1, 4) + d (4, 2) + d (2, 3) + d (3, 5)*
- *d (1, 5) = d (1, 2) + d (2, 4) + d (4, 3) + d (3, 5)*

This method is applied on the first 5 nodes in the segment-based chain. Then, the method is applied on the nodes from 2 to 6 in the chain. Afterwards, nodes from 3 to 7 are processed. The same mechanism keeps going forward until the last five nodes in the chain are processed.

3.2 Example

To further illustrate the SBTT chain formation algorithm with a detailed example, consider the post-order chain in the example in Fig. 1. Consequently, the SBTT algorithm initially starts with the list of vertices *POS* as follows: { *7, 11, 14, 1, 6, 0, 2, 3, 10, 5, 4, 13, 9, 8, 12* }. Next, a sequence of initial segments is constructed in phase 1 using Algorithm 1. The resulted segments at the end of phase 1 are: : { *11,* 7 }, { *14, 1* }, { *6, 0, 2, 3, 10, 5, 4* }, { *13, 9* } and { *8, 12* }. Then, segments { 11, 7 } and { 14, 1 } are concatenated (segment { 11, 7 } is reversed before concatenation because the distance between nodes 7 and 1 is the shortest among the end nodes of each segment).

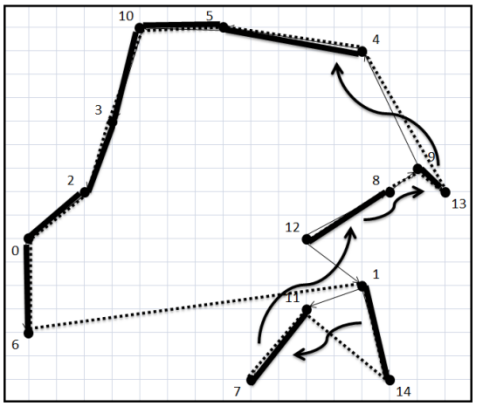

Figure 1. First phase of SBTT algorithm.

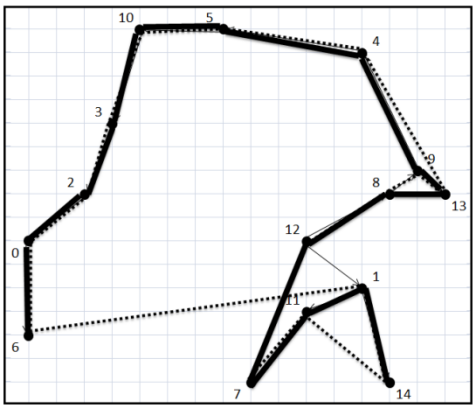

Figure 2. Comparison between post-order chain and final SBTT chain.

Then, segments { *6, 0, 2, 3, 10, 5, 4* } and { *13, 9* } are concatenated with the reversion of segment { *6, 0, 2, 3, 10, 5, 4* }. The resulted segments after this iteration are: { *14, 1, 11, 7* }, { *13, 9, 4, 5, 10, 3, 2, 0, 6* }, and { *8, 12* }. Afterwards, segments { *14, 1, 11, 7* } and { *8, 12* } are concatenated and none is reversed before concatenation. After this iteration, two segments are left: { *13, 9, 4, 5, 10, 3, 2, 0, 6* } and { *8, 12, 7, 11, 1, 14* }. Eventually, these two final segments are concatenated to produce the final single segment. Before concatenation, segment { *8, 12, 7, 11, 1, 14* } is reversed. The final single segment is: { *14, 1, 11, 7, 12, 8, 13, 9, 4, 5, 10, 3, 2, 0, 6* }.

The final SBTT chain after applying the modified 5-nodes version of Cross Elimination method is: { *14, 7, 11, 1, 12, 8, 13, 9, 4, 5, 10, 3, 2, 0, 6* }. This final segment is illustrated in Fig. 3.

3.3 Complexity Analysis

The complexity of running the segment-based algorithm goes as follows. First of all, it takes *O (N)* to execute phase 1 of the algorithm where *N* is number of nodes in the network. After phase 1 is completed, a number of *S* segments are constructed. Next, each segment is inspected and processed in order to determine which one of the other segments is the most suitable one to be concatenated with.

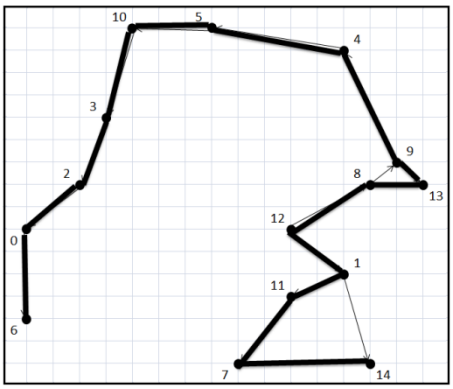

Figure 3. Segment-based chain after applying Cross Elimination method.

The total cost of doing this process for each segment is $O\ (S^2)$ where S is the number of currently existing segments. Furthermore, the last procedure gets repeated all over again until all segments are concatenated into one single segment; this is repeated p times where p keeps decreasing with each iteration until it equals 1. Additionally, it takes $O\ (N - 4)$ to run the Cross Elimination method. On the whole, the total cost of running the segment-based algorithm (Algorithm 1 and Algorithm 2) is:

$$N + (S^2)^p + (N-4) = O\ (N + (S^2)^p)$$

Nevertheless, it is important to mention that S is rather small compared to N. For instance, N=15 and S=5 initially in Fig. 2. Furthermore, S decreases after each iteration of p. In the previous example in sub-section *B*, S initially equals 5, but then it decreases to 3, 2, and finally to 1. Another important note is that p is also relatively small compared to N iterations and it is dependent on S; that is, when S=1 then p reaches its final iteration.

Finally, the total complexity of running the segment-based algorithm as a whole system is $O\ (N + (S^2)^p)$ in addition to the post-order complexity which is calculated in [10]. According to [10], it takes $O\ (N + E * log\ E)$ to construct the rooted directed minimum spanning tree where N is the number of nodes in the network and E is the number of edges. Moreover, it takes $O\ (N)$ to run the post-order tree traversal algorithm which yields the post-order chain.

4. Simulation Experiments and Results

To further achieve accurate simulations as much as possible, we have implemented greedy algorithm, post-order traversal and SBTT in C++ language using two simulating applications. First, we have implemented the algorithm using a network simulator developed in C++ by the authors of this research. Then, extended experiments have been carried out using QualNet simulator.

4.1 Simulation Environment

A token-passing approach has been used to run the PEGASIS data gathering protocol where it has been applied on the three studied algorithms (greedy, post-order and SBTT). The token-passing approach has been adopted in [10]. The simulations have been carried out on a Time Division Multiple Access (TDMA) system (also adopted in [10]). Sensor nodes are assumed to be capable of adjusting their transmission ranges depending on the distance to the receiving node [10][12]. Every sensor device in the network knows the physical coordinates of every other device. Furthermore, the highest-energy node selection strategy which is discussed in [5] is implemented for leader node selection.

In addition, we have assumed that the sink node is located at coordinates (50, 300) in a 100m x 100m network field [10][12]. In [10], 100-nodes have been located randomly in the network area; we have extended this to three categories: 10-nodes, 50-nodes, and 90-nodes. Moreover, packet size is set to 2000 bits and it does not change with data fusion, and the initial energy supplied to all sensor devices in the network is set to 1 Joule [10][12].

The radio model that has been used in our simulations is the same model used in [10][12]. In this radio model, the radio dissipates E_{elec} = 50 nJ/bit to run the transmitter or receiver circuitry, and ε_{amp} = 100 pJ/bit/ m^2 for the transmitter amplifier. To compute the transmission costs, the following formulas are used:

- Energy lost in transmitting a k-bit message over a distance d is given by: $E_{TX}\ (k,\ d) = E_{elec} * k + \varepsilon_{amp} * k * d^2$
- Energy lost in receiving a k-bit message: $E_{RX}\ (k) = E_{elec} * k$
- Energy lost in data fusion is 5 nJ/bit/message.

The main performance metrics that have been studied in this research are energy lost per round, network lifetime, and end-to-end delay.

There are three main performance metrics that have been studied in this research. The first one is the energy lost per round. It represents the average energy lost among all the nodes in the chain per round. This is an important performance metric that reflects the system's overall energy consumption. The second performance metric is the network lifetime. This is determined when the first node failure event occurs during the data gathering process. We measure this performance metric by the recorded number of successful rounds. The third and final performance metric is the end-to-end delay. We define this performance metric as the time that the data gathering process takes starting from the first data transmission (by one of the end nodes in the chain) until the leader node successfully transmits the aggregated data to the sink node.

We have conducted simulation experiments for these three main performance metrics in QualNet under three different values for N (where N is the number of nodes in the network field); these values have been increased from 10, to 50 and finally to 90.

Nevertheless, there are two important factors that directly affect these performance metrics; these factors are the average total length of the chain and the highest transmission range in the chain.

Naturally, the average total length of the chain is an essential factor that directly affects the overall energy consumption among the nodes. Needless to say, the shorter the chain is (physically speaking), the less energy it consumes. Additionally, the average total length of the chain is also an integral factor in the end-to-end delay performance metric. The highest transmission range in a chain is defined as the highest physical distance that exists

between two nodes in the chain. This metric is directly responsible for the network lifetime.

We have studied these two metrics under our developed C++ simulator by conducting 100 simulation trials for each value of *N* starting from *N*=10, *N*=11, *N*=12, ... , up to *N*=200.

4.2 Simulation Results and Discussion

1) Average Chain Length: Fig. 4 shows that the chain length in the three algorithms is relatively close for a small number of sensor devices. However, it can be observed from the figure that the difference between post-order chain length and the other two chain lengths increases and gets larger when number of sensor devices is increased.

One more observation is that our segment-based approach has a less chain length than the greedy approach; nonetheless, the difference is relatively small and they follow the same growth pattern. The increasing gap in chain length between post-order algorithm and the other two algorithms can be explained by inspecting the approach that each algorithm uses to traverse the nodes. If we go back to the definitions of the three approaches we find out that each one has a main traversing behavior as follows; greedy approach goes to the nearest node, post-order approach traverses sub-trees recursively in a left-most, root, and then right-most fashion, and SBTT goes to the nearest segment.

It is clear from the previous definitions that the greedy and SBTT approaches have something in common: location awareness. That is, the traversing process is done with respect to the nearest available units. On the other hand, post-order approach can be described as a blind traversal approach with respect to locations of nodes in the network field. The traversing process in post-order approach is carried out according to the order of the children of a certain node regardless of their physical locations in the network field. Additionally, cross elimination method has minimized the chain length in the simulation results for SBTT.

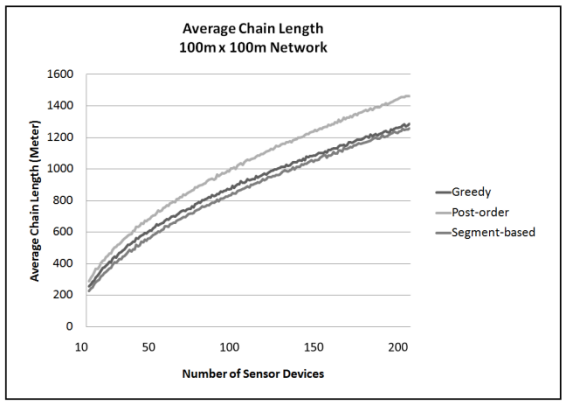

Figure 4. Simulation results for average chain length.

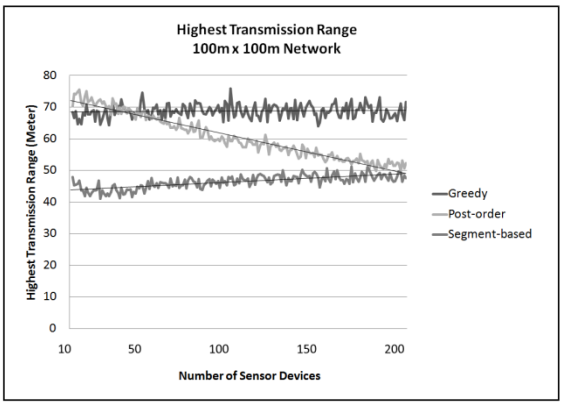

Figure 5. Simulation results for Highest Transmission Range (HTR).

2) Highest Transmission Range (HTR): The Highest Transmission Range (HTR) factor directly affects the network lifetime which we examine according to the first node failure occurrence in the network. Naturally, the node that is most likely to be the first node to encounter a failure due to energy depletion is the node that is losing energy faster than any other node in the network. In comparison with post-order traversal, segment-based traversal seems to have shorter and more balanced transmission ranges. For instance, if we compare the HTR in the example shown in Fig. 2 for post-order chain (edge E (1, 6)) with the HTR for segment-based chain (edge E (7, 12)), we will find out that there is an improvement in segment-based traversal with respect to this metric.

Fig. 5 shows that post-order approach records the highest values for HTR until around 40 nodes; SBTT approach records the lowest HTR values in this interval. After 40 nodes and up to 200 nodes, post-order approach starts recording lower values than greedy approach, and the values keep on decreasing until it records the same values of SBTT. Nevertheless, SBTT always records better values (shorter highest transmission ranges) than greedy approach at all value of N, and it also records better values than post-order approach up to 200 nodes.

The rapid improvement in post-order approach is due to the fact that post-order algorithm traverses the next left-most node (sub-tree) regardless of the physical distance between them. When the density of nodes is low in a network field, few sub-trees exist and these sub-trees are wide spread across the network field. However, the more nodes we have, the denser the tree becomes (sub-trees become closer to each other). As a result, the inefficient long distance traversals from root nodes to the next destination (next left-most sub-tree) is reduced. Moreover, SBTT algorithm is based on the minimum spanning tree topology which covers the shortest possible edges in the graph. This mechanism, in addition to minimizing the long inefficient communication ranges which used to result in post-order chain produces lower HTR values, thus, favoring SBTT in this field.

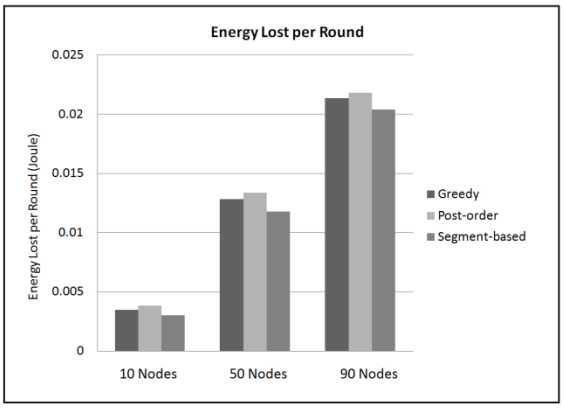

Figure 6. Simulation results for Energy Lost per Round (ELR).

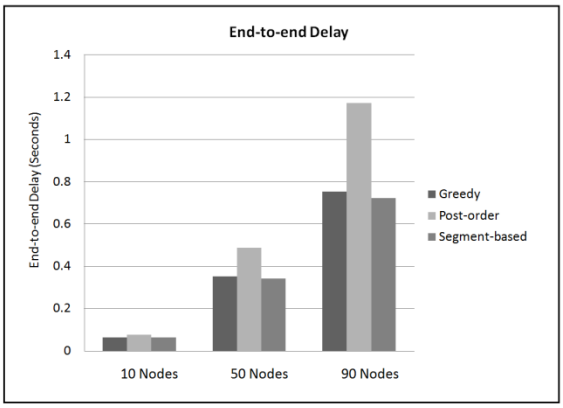

Figure 8. Simulation Results for end-to-end delay.

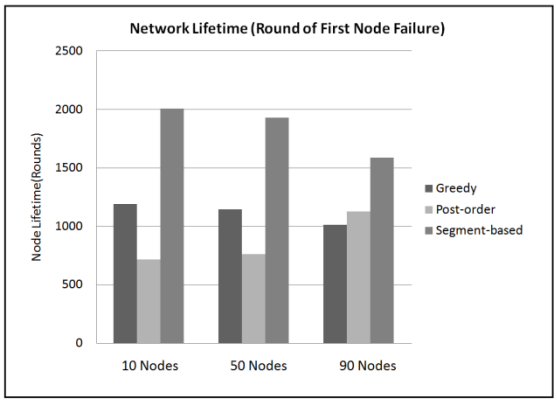

Figure 7. Simulation results for network lifetime.

3) Energy Lost per Round (ELR): Fig. 6 shows the results of our simulation experiments for Energy Lost per Round (ELR) performance metric. It can be concluded from the figure that post-order approach consumes more energy than the other two approaches. Moreover, our proposed segment-based approach consumes less energy than greedy approach. We believe that the chain length is directly related to the ELR results. Needless to say, the longer the chain is, the more energy it consumes.

The simulation results of the average chain length metric have shown that the chain that is obtained using post-order algorithm has the longest length among the three algorithms. The second longest chain is the one conducted using greedy algorithm, and finally, our proposed segment-based algorithm yields the shortest chain. These observations on the chain length results indeed match the ELR results that are shown in Fig. 6.

4) Network Lifetime: The simulations results for the network lifetime are shown in Fig. 7. At the first glance, we can observe that the simulation results are indeed directly related to the HTR metric. The results seem to be a reflection of the results for HTR. That is, high values of HTR produce opposite low values in the network lifetime at the same number of sensor devices, and vice versa. Our proposed segment-based approach has the longest network lifetime in low dense networks (up until approximately 200 nodes).

5) End-to-end Delay: Fig. 8 illustrates the simulation results that have been conducted for the end-to-end delay among the three studied approaches. According to Kurose and Ross [7], end-to-end delay can be calculated as indicated in (1), where N is the number of links, d_{proc} is the processing delay, d_{trans} is the transmission delay, and d_{prop} is the propagation delay.

$$d_{end-end} = N(d_{proc} + d_{trans} + d_{prop}) \quad (1)$$

Each delay is defined as follows. The processing delay is the required time to process the packets at each node. The transmission delay is the time that it takes each node to push the packet bits into the link. Finally, the propagation delay is the time that it takes each node to push the packet bits into the link.

Nevertheless, no queuing delay exists because of adopting the token-passing protocol. In our case, the processing delay is the same for each approach, and the transmission delay is equal among the three approaches too. However, the only delay factor that affects the overall end-to-end delay in each approach is the propagation delay.

Propagation delay is equal to d/s where d is the distance of the link and s is the wave propagation speed (which in wireless environment is equal to the speed of light). Therefore, the only factor that determines the difference in the end-to-end delay among the three approaches is the distances between the nodes, or more specifically the chain length.

The results show that end-to-end delay seems to increase when the chain gets longer. End-to-end delay is defined as the time that the data gathering process spends from the first transmission operation in the chain until the leader node delivers the final aggregated data to the sink node. Therefore, this performance metric represents the

time that the protocol spends to execute a successful round, hence, it is directly affected by the length of the chain.

As a result, post-order approach has been observed to yield the longest end-to-end delay values. Moreover, the more sensor devices are added to the network the more end-to-end delay is recorded with the post-order approach. On the other hand, our proposed segment-based approach produces the least end-to-end delay values; this is due to its capability of producing chains with the least lengths among the three algorithms. The greedy chain produces higher end-to-end delay than the segment-based chain. One last note is that the difference in end-to-end delay between the post-order approach and the other two algorithms gets gradually higher with the increase in number of sensor devices in the network.

Table I. presents the percentages of improvements in segment-based approach over the greedy and post-order approaches at 90 nodes for the studied performance metrics.

TABLE I. PERCENTAGES OF IMPROVEMENT IN SEGMENT-BASED APPROACH OVER GREEDY AND POST-ORDER APPROACHES AT 90 NODES

Metric	Improvement over Greedy Approach	Improvement over Post-order Approach
Chain Length	5%	16%
HTR	30%	21%
ELR	1%	2%
Network Lifetime	57%	41%
End-to-end Delay	9%	43%

5. Conclusion

This research paper presents a Segment-Based Tree Traversal algorithm (SBTT). It is a location-aware algorithm that is used to construct an improved chain for PEGASIS protocol in WSNs. The chain which is constructed using SBTT algorithm has shorter physical distances among its nodes and has lower HTR values. These qualities have in turn enhanced the overall data gathering process, thus, improving the overall network performance. The proposed approach has slightly improved the energy lost per round compared to the greedy and post-order approaches in chain construction. Additionally, it has improved the network lifetime and the end-to-end delay.

References

[1] D. I. Sacaleanu, D. M. Ofrim, R. Stoian, V. Lazarescu, "In node data processing for increasing the lifetime of a wireless sensor network", Proceedings of the 5th WSEAS International Conference on Sensors and Signals (SENSIG '12), Sliema, Malta, September 2012, Advances in Sensors, Signals, Visualization, Imaging and Simulation ISBN: 978-1-61804-119-7 WSEAS Press 2012.

[2] E. De Poorter, S. Bouckaert, I. Moerman, and P. Demeester, "Non-intrusive aggregation in wireless sensor networks", Elsevier B.V. Journal, E. De Poorter et al. / Ad Hoc Networks 9 (2011). J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] H. Alhasan, M. Qatawneh, A. Sleit, W. Almobaideen, "EAPHRN: energy-aware PEGASIS-based hierarchal routing protocol for wireless sensor networks", Journal of American Science, 2011.

[4] H. Min, S. Yi, J. Heo, and Y. Cho, "Energy-efficient data aggregation protocol for location-aware wireless sensor networks", International Symposium on Parallel and Distributed Processing with Applications 2008, IEEE DOI 10.1109/ISPA.2008.38.

[5] I. Shukla and N. Meghanathan, "Impact of leader selection strategies on the PEGASIS data gathering protocol for wireless sensor networks," Ubiquitous Computing and Communication Journal, vol. 4, no. 5, December 2009.

[6] J. Gómez, R. Poveda, and E. León, "Grisland: a parallel genetic algorithm for finding near optimal solutions to the travelling salesman problem", GECCO '09 Montréal Québec, Canada, ACM 978-1-60558-505-5/09/07.

[7] J. Kurose and K. Ross, "Computer networking: a top-down aproach", 5th edition, Pearson Press, ISBN-10: 0-13-136548-7, 2010-02-08.

[8] K. Patel, C. Patel, S. S. Rizvi, K. M. Elleithy, "An efficient approach to reduce energy consumption in wireless sensor networks through active nodes optimization", URI - NE ASEE 2007 Conference.

[9] M. A. Jassim, "A multi-function energy-efficient system model for location-aware wireless sensor networks", master's thesis, Department of Computer Science, The University of Jordan, Amman, December 2011.

[10] N. Meganathan, "Use of tree traversal algorithms for chain formation in the PEGASIS data gathering protocol for wireless sensor networks", KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS VOL. 3, NO. 6, December 2009.

[11] N. S. Patil and P. R. Patil, "Data aggregation in wireless sensor network", 2010 IEEE International Conference on Computational Intelligence and Computing Research.

[12] S. Lindsey and C. S. Raghavendra, "PEGASIS: power-efficient gathering in sensor information systems", Aerospace Conference Proceedings, 2002. IEEE.

[13] S. Rachamalla, A. Sheela, "Routing protocols for wireless sensor networks – a survey", Proceedings of WSEAS International Conferences in Cambridge, MA, USA, January – February, 2013, Recent Advances in Electrical and Computer Engineering, ISBN: 978-1-61804-156-2 WSEAS Press 2013.

[14] W. Almobaideen, "SPDA: stability based partially disjoint AOMDV", European Journal of Scientific Research, vol.27 No.3, pp.342-348, 2009.

[15] W. Almobaideen, D. Alkhateeb, A. Sleit, M. Qatawneh, K. Qadadeh, R. Alkhdour, H. Abu Hafeeza, " Improved stability based partially disjoint AOMDV", Int. J. Communications, Network and System Sciences, doi:10.4236/ijcns.2013.65027, May 2013.