Abstract—Placement of nodes in Wireless Sensor Network (WSNs) is often massive and random. Topology control algorithms focus in lowering the initial network topology, by reducing active nodes and links, thus saving resources and increasing network lifetime. Currently, most algorithms and schemes in WSNs, construct shared, core-based trees, with the sink as a root for this purpose. In this paper, we study whether trees that initiate from each source, called source-based trees, can assist in this purpose and provide, under specific circumstances, an efficient topology control solution. Simulation results show that source-based trees can provide proper topology and routing control solution in WSNs, when are employed along with “resource”, congestion control algorithms in WSNs.

Index Terms—Wireless Sensor Networks, Congestion Control, Source Based Trees, Sink Based Trees

I. INTRODUCTION

Massive and random placement of sensor nodes on a monitored field renders node communication a difficult task to be achieved. Interference, congestion, and routing problems are possible to arise at any point in such networks. Routing challenges in WSNs stem from the unique characteristics of these networks, such as limited energy supply, limited computing power, and limited bandwidth on the wireless links, which impose severe restrictions on the design of efficient routing protocols. According to [1], a number of routing challenges and design issues like, among others, node placement and energy consumption, can affect routing process in WSNs. Thus, topology control, in conjunction with routing challenges, becomes an important issue that has to be carefully considered in order to achieve proper network operation.

Generally, congestion control algorithms in WSNs employ two methods in order to control and avoid congestion [2]. The first method is called traffic control and the second resource control. Algorithms that employ the traffic control method, adjust the rate with which sources inject traffic to the network in order to control congestion. On the other hand, resource control algorithms, employ redundant nodes, which are not in the initial path from source to sink, in the process of forwarding data. Thus, algorithms that employ this method do not control the data rate of the sources but the paths though which the data flows. According to studies [3] [4], traffic control algorithms are not affected by different node placements, while according to the same studies resource control algorithms are significantly affected. Different node placements create a variable number of paths which are important for the proper operation of these algorithms.

Placement of nodes in Wireless Sensor Network (WSNs) is often massive and random. Permitting all nodes to transmit concurrently without any control will result in high interference, high energy consumption, and reduced network lifetime. Topology control algorithms focus in lowering the initial network topology, by reducing active nodes and links, thus saving resources and increasing network lifetime. Employing tree structures as topology control algorithms in WSNs has been widely adopted, since it is an efficient and robust solution. The overwhelming majority of algorithms in WSNs construct shared, core-based trees, with the sink as a root. On the other hand, trees that initiate from each source, called source-based trees, are avoided, since each source must create its own routing tree, a fact that creates significant overhead to the network.

In this work we study and show through simulations, that source based trees can provide a proper topology and routing control solution in WSNs, if employed under specific circumstances and for specific applications. In particular, we study the influence of source-based trees when they serve as topology control schemes in “resource control” congestion control algorithms.

II. ANALYSIS

A shared, core-based tree is an efficient topology control solution in WSNs. Usually, such trees are build as spanning trees using the sink as the root and all nodes forward their data using this structure. Using the sink node as the root is optimum, since the sink in WSNs is, most of the times, a robust node that does not suffer from power limitation issues. Thus, this situation defeats the main disadvantage of shared, core-based tree, which is the fact that core nodes can become single points of failure. The creation of a sink-based tree is normally a simple procedure that begins after the network discovery phase. Nodes become part of this tree at the initialization of the network and maintain their position in it until topology changes happen.

In this work we consider that the following assumptions exist:

Source-Based Routing Trees for Efficient Congestion Control in Wireless Sensor Networks

Charalampos Sergiou and Vasos Vassiliou
Networks Research Laboratory
Department of Computer Science
University of Cyprus
Nicosia, Cyprus
Email: {sergiou,vasosv}@cs.ucy.ac.cy
- Nodes communicate only with nodes that are one hop away
- Dense placement of nodes achieves the coverage of the network
- As we describe in the next subsection, nodes route packets only to the nodes that are one level higher than themselves

Many wireless sensor network applications rely on the availability of a collection service to route data packets towards a sink node. A typical collection protocol provides for the construction and maintenance of one or more routing trees having each a so-called sink node as their root. A sink can store the received packets or forward them to an external network, typically through a reliable and possibly wired communication link. Within the network, nodes forward packets through the routing tree up to (at least) one of the sinks. To this end, each node selects one of its neighboring nodes as its parent. Nodes acting as parents are responsible of handling the packets they receive from their children and forwarding them towards the sink. To construct and maintain a routing tree a collection protocol must first define a metric to be used by each node to select its parent. The distance in hops to the sink or the quality of the local communication link (or a function thereof) can for instance be used as metrics for parent selection. In either cases, nodes need to collect information about their neighboring nodes in order to compute the parent selection metric. To this end, nodes regularly exchange corresponding messages, usually called beacons, that contain information about, e.g., the (estimated) distance in hops of the node to the sink or its residual energy.

A. Sink-Based Tree Creation

The first phase of network establishment is network discovery. In this phase, the major task of every node is to discover its neighbor nodes in the network. A common method for network discovery is some sort of flooding algorithm, initiated and controlled by the sink. After the end of this phase each node maintains a routing table in which the IDs of the neighbors nodes are kept, as well as any other information requested by the algorithm’s designer. An example of the end of initial phase is presented in Figure 1 in which the nodes are aware of all of their one-hop neighbors.

It is clear that the type of initial topology shown in Figure 1 must be reduced, otherwise interference and energy issues are possible to arise. For example, if an event appears and is captured by nodes 10, 11, and 20 and all three nodes start transmitting packets to all the nodes they are connected with, then it is certain that after some time, interference and buffer based congestion will arise. Congestion will lead to quick power exhaustion and the network will not be able to accomplish its task. Thus, the network initial topology must be reduced in order to take into account the redundancy in the connectivity.

Phase 2 of the tree creation begins from the sink and the target is to reduce the initial topology and place nodes in levels from the sink to the edge nodes. The process initiates with the sink broadcasting a “Hello” message with the level field set to 0. Nodes that receive this message update their level field to 1 and re-broadcast this message. Each node that receives a “Hello” message, updates its level field and broadcasts this message accordingly. If a node receives more than one “Hello” messages, it updates its level field based on the lower level. Thus, if its level changes to a lower value, it informs the nodes around it and the process iterates until all nodes are set in levels. Hence, a spanning tree is created.

To explain this concept better let us consider again Fig.

1. The sink broadcasts a “Hello” message with level set to 0. Nodes 1, 2 and 3 receive this message, update their level field to 1, and re-broadcast this message. Nodes that receive the message from level 1 nodes, also update their level and the procedure iterates until all nodes connect at the lowest level possible. Fig. 2 presents the results after this procedure is applied to the network.

As shown in Fig. 2, creating a level-based, shared tree, leads to a “relaxed” topology (with a lower degree of connectivity). Nodes communicate with a lesser number of nodes, a fact that renders the network more capable to avoid and control congestion situations (either in the medium or in the buffers) while the source-to-sink packet time is reduced to the minimum, since a spanning tree is created.

B. Source-Based Tree Creation

Source-based trees introduce a completely different routing concept in comparison with sink-based tree. The biggest difference lies on the fact that source-based trees, abandon the shared, core-based, philosophy and move to distributed solutions. Source-based trees are constructed on demand (when a node becomes a source node) and continue their operation until nodes stop being sources. Hence, when a node senses that is becoming a source node (receives analog data) it starts building its routing tree from itself (source) to the sink.
Building a proper and well tuned routing tree from a source to the sink is not a straightforward procedure as with shared, core-based trees. The reason lies on the fact that source-based trees do not "reduce" the topology as shared, core-based trees do. On the contrary, they incur significant overhead since every node is possible create its own tree, due to the fact that every node can become a source.

A “naive” approach, is to construct a source-based tree just based on level information (similar to the sink-based solution). For example, if we use the topology of Fig. 1 this will result in the tree of Fig. 3. This tree is considered “naive”, since it presents several drawbacks that impose severe restrictions to the proper operation of the network.

One of the drawbacks is that it does not create unique trees using the shortest path to the sink. We also notice that it is possible for a packet to be routed to a node that does not have any higher level node (closer to the sink). For example, if a packet is forwarded to node 14 then this node is not possible to know to which node to forward this packet to. Node 14 is connected to node 7 and 20, which are at the same level and node 11 and 19 which are at a lower level. Thus, if packet is forwarded to node 20 (which is at the same level as node 14) then node 20 will come to the same situation as node 14. Routing loops is possible to be created between these nodes.

Thus, source based trees presents several disadvantages that have to be addressed before they get used in topology control and routing.

III. SOURCE-BASED TREES IN CONGESTION CONTROL ALGORITHMS

A normal question that arises, is why to employ source based trees for any reason, since sink-based trees are efficient while source-based trees present several disadvantages. Before answering this question we shortly present the ways that congestion is controlled in WSNs.

Generally two methods exists with which algorithm designers attempt to control congestion in WSNs. The first method is called “traffic control”. Algorithms that employ the traffic control method, adjust the traffic that is injected to the network by the different sources, in order to cope with the capacity of the currently employed paths. Thus, the rate of the sources is reduced until congestion is alleviated. This method presents similarities with the traditional congestion control algorithms in wired networks and it has also been adopted by the large number of congestion control algorithms in WSNs [5], [6], [7], [8].

The second method is called “resource control”. This method, the network takes advantage of the redundant deployment of sensor nodes in the field and employs the resources (buffer, power) of nodes that do not participate in the initial routing paths to forward data through them. Thus, sources do not reduce the rate with which they inject packets in the network and excess packets are forwarded through alternative or multiple paths [2], [9], [10].

Recent research work [3] and [4] has shown that traffic control algorithms are not significantly affected by different node placements. On the other hand resource control algorithms are affected by different placements. We believe that properly employed topology control algorithms are able to provide additional benefits to resource control algorithms. Source-based routing trees provide numerous alternative paths, in comparison with sink-based trees, which, if carefully selected, can increase the performance of such algorithms.

A. Issues of Source-Based Trees

To construct source-based trees which can fulfill their purpose we consider a small set of “critical” parameters:

Location Awareness - Nodes must be aware of their position in relation to the sink. Localization and positioning in WSNs is a subject that attracted a lot of research work [11] and solutions already exist in the literature. Hence, if a node
is aware of its position then it selects as neighbors only nodes that are closer to sink than itself. For example in Fig. 3 node 27 will not be selected by node 25, due to its position which is further away from the sink, even though it is a level 2 node.

**Higher Level Connection Availability** - A node is kept on the tree only if it is connected to at least one node at a higher level than itself. For example in Fig. 3 node 14 will remain out of the tree since it is not connected to a node at a higher level. Concerning this parameter, an enhancement could be the connection with nodes at the same level, but closer to the sink. In this work we examine just the influence of critical parameters, and we avoid to involve any enhancements.

**Number of nodes kept in neighbor table** - In dense placements this number could be very big and will create significant overhead problems to the network. Thus, a maximum number of neighbors must be set. According to [12] a value of six neighbors seems to be optimal. Also a minimum number of two must exist. If a node is connected to less than two higher level nodes, it must add in its neighbor table nodes which are at the same level, but nearest to the sink. If such nodes do not exist, then this node should not be considered as a good neighbor for data forwarding.

Employing these parameters in Fig. 3, the initial source based tree transforms to the one in Fig. 4.

![Fig. 4. A properly constructed source based tree](image)

The resulting topology is significantly reduced in comparison with Fig. 3 while all nodes are connected to upstream nodes and each node provides at least two alternative paths. Nodes that do not comply with the parameters discussed above, are not part of this routing tree. Of course, due to the dynamic nature of WSNs, is possible that a node which is not at the routing table the present moment, to become part of it, in case of topology changes.

Also, although Fig. 4 seems even simpler than sink based tree topology (Fig. 2), sink based tree topology is a global, shared topology while source based tree (Fig. 4) concerns just one source node, out of possible, all nodes in the network. This means that if many sources exist in the network there could also exist many different trees that share the same nodes.

### IV. Evaluation

To compare the performance of source based against sink based trees, concerning resource control algorithms for congestion control in WSNs, we performed a series of simulations.

#### A. Network Model

We assume that there is a WSN, where nodes are initially deployed uniformly. One sink is located at a specific point in the network. Each node is considered that is aware of its position in relation to the sink. In our network deployment we consider that all nodes (except the sink) are identical and “carrier sense multiple access with collision avoidance” (CSMA/CA) is employed as MAC protocol.

Moreover, in order to compare the performance of source and sink based trees when the network faces congestion, we employed as a resource control reaction mechanism, the alternative path creation scheme of Hierarchical Tree Alternative Path (HTAP) algorithm [9]. HTAP is a resource congestion control algorithm in WSNs and consists of four schemes:

- Topology control
- Hierarchical Tree Creation
- Alternative Path Creation
- Handling of Powerless (dead nodes)

We stress that we have employed just the alternative path creation scheme, which is described below.

Alternative path creation runs when the buffer of a node starts filling with a rate with which buffer occupancy is increasing. In this case each node runs locally a lightweight congestion detection (CD) algorithm. When the buffer occupancy reaches a specific value, CD algorithm starts counting the rate with which packets are reaching the node. Since each packet is identified by the NodeID in its packets header, CD algorithm is aware of the nodes that are transmitting packets through this node, as well as, for their data rate. Thus, CD algorithm using Equation 1

\[
\sum_{i=1}^{k} R_{x_i} \geq T_{x_{max}}
\]

is able to calculate the total receiving rate and compare it with the maximum rate that is able to transmit. When this rate exceeds this threshold, node is sending a backpressure message to specific nodes that are transmitting packets through it. The selection of these node is performed firstly from the node(s) that transmit with the lower rate. The purpose of this tactic is to maintain the throughput of nodes to the maximum possible level without packet drops, in order to maintain the performance characteristics of the network, since all nodes initially use the shortest path to sink.

Thus, when a node is informed through the bi-directional link to stop transmitting through a specific node, it searches in its neighbor table and finds the next available node with the next smaller or even same level (in comparison with the
congested node) and start transmitting through it. If this node is in the transmitting range of the congested node is already aware of its condition or if it is not in its transmission range it is not possible to route packets through it again, since all links are bi-directional. Thus, the congestion hotspot in this phase is avoided.

B. Simulator Setup

For our simulations we employed the Prowler simulator [13]. The radio propagation and transmission models are given by:

\[ P_{rec,\text{ideal}}(d) \leftarrow P_{\text{transmit}} \frac{1}{1 + d^{\gamma}} \]  

where, \( 2 \leq \gamma \leq 4 \) and

\[ P_{rec}(i,j) \leftarrow P_{rec,\text{ideal}}(d_{i,j})(1 + a(i,j))(1 + \beta(t)) \]

where \( P_{\text{transmit}} \) is the signal strength at the transmitter and \( P_{rec,\text{ideal}}(d) \) is the ideal received signal strength at distance \( d \), \( a \) and \( \beta \) are random variables with normal distributions \( N(0, \sigma_{a}) \) and \( N(0, \sigma_{\beta}) \), respectively. A node \( j \) can receive packets from node \( i \) if \( P_{rec}(i,j) > \Delta \) where \( \Delta \) is the threshold.

In our simulations we use the following default simulator parameters:

- \( \sigma_{a} = 0.5 \)
- \( \sigma_{\beta} = 0.03 \)
- \( p_{\text{error}} = 0.05 \)
- \( \Delta = 0.1 \)

The rest of the parameters reflect Mica-Z node specifications, the most important of which are presented in Table I.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>SIMULATION PARAMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Data Rate (kbps)</td>
<td>128</td>
</tr>
<tr>
<td>Transmission Power (dbm)</td>
<td>2</td>
</tr>
<tr>
<td>Receive Threshold (dbm)</td>
<td>-74</td>
</tr>
<tr>
<td>Transmission Current (mA)</td>
<td>17.4</td>
</tr>
<tr>
<td>Receive Current (mA)</td>
<td>19.7</td>
</tr>
<tr>
<td>Fragment Size (bit)</td>
<td>1024</td>
</tr>
<tr>
<td>Buffer Size(Bytes)</td>
<td>512K</td>
</tr>
<tr>
<td>MAC layer</td>
<td>CSMA/CA</td>
</tr>
</tbody>
</table>

C. Scenarios and Results

For each of the performance metrics presented below, the results are the average of 20 runs for 10s for each measurement point (except for the cases that it is otherwise stated). An initial number of 100 nodes, are uniformly placed on a square grid over an area of 500x500m. All nodes communicate only with the nodes that are one hop away. In this simulation series we have programmed every source to transmit constantly 128 packets/s while we increase the number of sources. Using these simulation settings we expect to analyze the performance of the network when the data load is increasing.

The first metric we examine is sink throughput. Sink throughput indicates the ability of any congestion control algorithm, to control congestion and transmit to sink a maximum number of packets. In our study, since the congestion control mechanism is identical in all three cases we examine, results provide an indication of how a resource control algorithm is affected by different topology control schemes.

![Fig. 5. Sink Throughput](image)

Results in Fig. 5 indicate that as the number of sources in the network increases, the source-based tree topology is more able to increase the sink throughput compared with the sink-based topology. The reason lies on the fact that when the sources increase, the sink-based topology, which is a shared topology, provides a limited number of alternative paths to each node. Thus, when congestion occurs and the capacity of the routing tree is exceeded, alternative paths are limited and the throughput reduces. On the other hand a “naive” source-based tree, as it is expected, is not able to provide the sink with an acceptable number of packets. Routing circles and uncontrolled number of alternative paths enhance congestion problem instead of relieving it.

The next parameter we study is the average delay of successfully received packets from the sources to the sink. We choose to count the delay of successfully received packets since we focus on the efficiency of topology control algorithms, in terms of alternative path creation, and not on the efficiency of the congestion control algorithm.

Results show that as the number of sources in the network increases, the average delay increases in all three cases. This happens because congestion occurs and the alternative path creation algorithm routes the excess packets from different paths which are longer that the initial paths (the initial path is always the shortest from sources to sink). Comparing the three schemes we notice that sink-based trees generate the less delay. These results are in agreement with the results of Sink throughput (Fig. 5), since the routing tree that is created is a spanning tree and the position from each node to the sink is the minimum. Source-based trees provide longer delays than sink-based trees. The reason is the bigger number of paths generated, which are not always the shortest to the destination. On the other hand the “naive” source-based tree provides the
worst results, since in such case a packet can follow very long routes in order to reach the sink.

Finally, we study the average energy consumption of the nodes, within a specific period of time. This period of time is the time before a single node in the network gets power exhausted. We make this choice because no topology maintenance algorithms are incorporated in the algorithms we examine, therefore until this event all algorithms are operating under the same conditions. The used energy consumption of a node is the energy used for communication, including transmitting (Pt), receiving (Pr), and idling (Pi). We measure the power consumed in units where each unit equals to 1000mW in 1s (W x s).

Results depict that sink-based tree provides the better energy consumption in comparison with the other schemes. The source-based tree presents slightly worse results compared to the sink-based tree and both present much better results than “naive” source-based tree. Although it could be expected that based on Fig.6, sink-based trees could have better energy consumption, the fact that they cannot handle so efficiently the heavy data load, results in some packet drops and energy waste.

V. CONCLUSIONS

In this work we study tree-based topology control algorithms in WSNs. We present source- and shared-based trees and how they apply in a randomly deployed WSN. Then, using simulations we compare the performance of the two topology control schemes as well as the “naive” source-based tree when a resource control algorithm applies. Simulation results show that source-based trees could be more efficient when the data load is heavy, since they provide more routing paths from each node. On the other hand, sink-based trees provide better results in terms of delay and energy consumption. The major conclusion that we extract from this study is that source-based trees, if carefully tuned, can provide an efficient topology control solution for specific applications.

ACKNOWLEDGMENTS.

This work has been partly conducted under the European Union Project GINSENG funded under the FP7 Program (FP7/2007-2013) grant agreement no 224282.

REFERENCES