Using a Distributed Snapshot Algorithm in Wireless Sensor Networks

Ana Paula R. da Silva
anapaula@dcc.ufmg.br

Fernando A. Teixeira
teixeira@dcc.ufmg.br

Rafael Kelles V. Lage
kelles@dcc.ufmg.br

Linnyer B. Ruiz
linnyer@dcc.ufmg.br

Antonio A.F. Loureiro
loureiro@dcc.ufmg.br

José Marcos S. Nogueira
jmarcos@dcc.ufmg.br

Federal University of Minas Gerais
Department of Computer Science
Belo Horizonte, Minas Gerais, Brazil

Abstract

Wireless Sensor Networks (WSNs) have particular characteristics that do not allow to apply traditional distributed algorithms directly to them. In this work we adapt the algorithms Distributed Snapshot, Broadcast and Propagation of Information with Feedback (PIF) to WSNs and apply them to generate the Energy Map of a WSN. This map shows the behavior of such a network and can be used to predict its behavior. We simulate the algorithms proposed and show their number of messages, energy spent and execution time.

1. Introduction

A Wireless Sensor Network (WSN) is a new type of wireless communication network that can be applied in a variety of scenarios such as environment monitoring, security, and medical applications. Sensor nodes in WSNs are spread in a region to be sensed and they communicate among themselves using wireless point-to-point communication forming an ad-hoc network. Sensors collect, process and send data from the environment to others nodes. Basically, there are three different types of nodes: common nodes that are responsible for collecting sensing data, sink nodes that are responsible for receiving, storing and processing data from common nodes and “gateway” nodes that connect the sink nodes to external entities called observers (final users). A wireless sensor node is composed by memory, processor, transceiver, battery, and one or more sensors, that can collect different data types such as temperature, pressure, electromagnetic field, and chemical agents. A sensor node can run a program according to its hardware restrictions.

A WSN is different from a traditional computer network in several aspects. Usually a sensor network consists of a large number of distributed nodes, have energy restrictions, and must have mechanisms for self-configuration and adaptation in case of a communication failure and node loss. It means that traditional distributed algorithms, as a communication protocol and leader election, must be adapted for this kind of environment before being used. An important aspect in WSNs is to monitor the available energy in the network since, in general, a node battery is neither replaced nor recharged. It means that when the battery runs out the node becomes unavailable.

In this work, we propose using a variation of the Distributed Snapshot Algorithm (DSA) [1] together with other adapted algorithms, to obtain the global state of a WSN. This global state was used to build the energy map, but can be used for other applications as well. In the energy map we can observe the energy decreasing, and, then, foresee which nodes will become unavailable. With this information, it is possible to do a better network configuration, or even to intervene in an already installed network, to extend its life time. The algorithms proposed here are for a “flat” network – where there is no grouping or hierarchy and “homogeneous” – where all nodes have the same capability of sensing and processing, as depicted in Figure 1.

The rest of this paper is organized as follows. Section 2 describes the Distributed Snapshot Algorithm and the challenges for its utilization in WSNs. Section 3 presents the proposed modifications for the DSA algorithm and for the broadcast and PIF (Propagation of Information with Feedback) [4] to obtain the network global state. Sections 4 and 5 present the simulations and results respectively. Finally, Section 6 presents our concluding remarks.

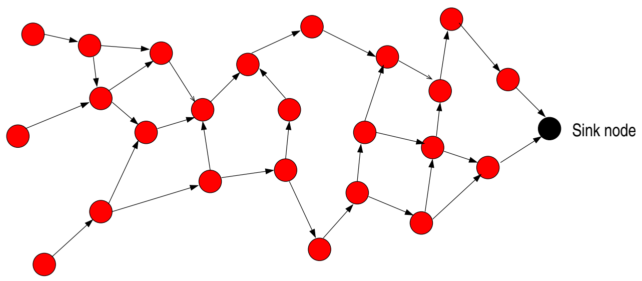

Figure 1. An example of a flat and homogeneous WSN showing possible paths from common nodes to a sink node.

2. Distributed Snapshot Algorithm for WSNs

The Distributed Snapshot Algorithm obtains a distributed system global state, that is, it records the consistent state in a given moment. The algorithm deals with two independent distributed computations. The first computation refers to the substrate, i.e., the system properties one wishes to study, and the second one refers to the snapshot algorithm itself. The only interaction between the two computations is that the computation of the snapshot algorithm is able to get information about the substrate in order to record the global state of the system.

Some of the challenges when applying the Distributed Snapshot Algorithm to obtain information from the sensors in a WSN are energy consumption, dynamic topology, the algorithm to collect the local states, and network fragmentation.

Energy consumption is a critical problem in WSNs since sensors depend on a battery source to work. In general, a WSN is designed to last for as long as possible. Any activity performed by a sensor consumes energy. In general, the transmission/reception of a data spends more energy than its processing. One goal of this work is to measure the energy spent by the distributed algorithm itself when executing a computation in a sensor node.

In a WSN, the topology tends to be dynamic even if the nodes are fixed. The topology can change because nodes can become unavailable temporarily or definitely. A node is temporarily unavailable when there is a communication problem or becomes suspended (inactive). A node is definitely unavailable when its battery runs out or it is destroyed. These situations can affect the behavior of the DSA since a node can only conclude its local snapshot when it receives markers from all its neighbors.

After the local state is recorded, it is necessary an algorithm to collect these local states to obtain the global state of the network (the network energy map in our application). The use of this algorithm in WSNs is a challenge as much as the DSA is since it must consume as little energy as possible.

The network fragmentation is closely related to the dynamic topology. If a node becomes unavailable, the network may split and one part will no longer reach the other. This is also a problem to the DSA. Furthermore, it raises a conceptual question: if there is an unreachable area, does it make sense to keep trying to take snapshots of that region? The answer probably depends on the underlying application.

3. Proposed algorithms

This section presents the proposed algorithms DSA_WSN, Broadcast_WSN, and PIF_WSN, that are versions of DSA, Broadcast, and PIF for a WSN, respectively. The original PIF (Propagation of Information with Feedback) algorithm [4] builds a spanning tree having the sink as the root.

Before discussing the energy spent by the DSA, Broadcast and PIF algorithms, we must make sure that these algorithms will work correctly in a WSN. Considering the problems discussed in the previous section, we present the strategies applied to the DSA to continue working despite the problems raised by the dynamic topology and network fragmentation. We will also discuss the problems and advantages of using the Broadcast and PIF algorithms to collect the local states. The energy spent by these algorithms will be discussed in Section 5.

3.1. Dynamic topology

The sensors can become unavailable and the algorithms mentioned above were not initially designed to cope with this situation. Note that sensors can become unavailable in different moments during the algorithm. Depending on the moment this situation happens, it can be treated by the algorithm in different ways.

There are three cases when a node becomes unavailable. A node can become unavailable before the start of the DSA, during the snapshot step and during the collecting step using Broadcast or PIF. In particular, during the snapshot step, the node can become unavailable in three different moments: after receiving a marker and before sending this marker to another node; after sending the marker and before recording the local state; and after recording the global state. In all these cases, the DSA will not be concluded. To make the Distributed Snapshot Algorithm fault-tolerant, we have created a special message called "death warning". A node sends the death warning to all its neighbors when it is close to become unavailable. Depending on the moment that a node becomes unavailable, the "death warning" can contain the local state of the sensor. In this work we assume that the node will become unavailable only due to lack of energy.

In our implementation, we have assumed that each node knows the number of its neighbors. In our fault model, we have assumed that messages are not lost. Of course, there are situations that these propositions do not happen. However, it is certainly possible to imagine scenarios that is possible to make these suppositions, at least with a very high probability.

The DSA_WSN algorithm (DSA modified) works as follows. If a node dies before the beginning of the snapshot its neighbors will interpret its death warning only as they have one neighbor less. The snapshot algorithm will work as usual. If the node dies during the snapshot algorithm and its neighbors are waiting for a marker, then when a neighbor receives a death warning it will remove the node that has died from its list of neighbors and will decrement the number of missing markers. Then it will go on with recording its own local state. If the sensor dies after it has sent the marker, the neighbor will remove this node from its neighbors' list. If the node dies after it has recorded its local state, it will send this state (snapshot) with the death warning, therefore this information will not be lost with the node death. If the node dies during the algorithm that collects the local sates, the death warning will contain the local state of the dead node and its neighbors will take it out from their neighbors' list.

The problem will be if the node is not able to send the death warning. It can happen, for example, when it is ready to send the death warning but it receives so many messages that its energy is not enough to send it. To avoid this possibility, we assume that the sensor will always have the secure level of energy, enough for it to send the death warning notwithstanding the number of messages that it receives. With these adaptations, the modified DSA can work in wireless sensor networks even if the network splits.

Figure 2 presents the pseudo-code of the DSA_WSN algorithm adapted to WSNs.

3.2. Local states collecting and network fragmentation

Our second problem is how to collect and assemble the local states, that is, the local snapshots, to have a global state. In this work, the global state corresponds to a map of the energy level in each node in the network in a given moment. We will call this map as "energy map". We took as baseline an algorithm based on the broadcast. When each node has concluded its own snapshot, it sends a broadcast message with its local state (snapshot) to all its neighbors. As one node receives the broadcast message with the snapshot from a neighbor, it forwards this message to all its neighbors as well. This algorithm finishes its computation when all nodes have received snapshots from the whole network. The question is that only the sink needs to know about the entire network. But as we consider the sink to be fixed, we use the broadcast to convey the information to it. We have used three different topologies in our simulation in order to evaluate the energy consumption in getting the global snapshot. These topologies will be shown in Section 4.

```
ALGORITHM DSA_WSN:
 1. Sink node sends a ⟨MARKER⟩ msg;
 2. if node $n$ receives ⟨MARKER⟩ msg then
 3.   begin
 4.     if node $n$ is not in SNAPPING mode then
 5.       begin
 6.         Change the local state to SNAPPING;
 7.         Record state;
 8.         Forward ⟨MARKER⟩ msg;
 9.       end;
10.     if returning ⟨MARKER⟩ then
11.       Decrease the number of waiting ⟨MARKER⟩s;
12.     if there is no more returning ⟨MARKER⟩s to receive then
13.       Local state is ready;
14.   end;
15. if node $n$ receives msg ⟨DEATH_NOTIFICATION⟩ then
16.   begin
17.     Mark neighbor as dead;
18.     if node $n$ is in SNAPPING mode then
19.       Treat ⟨DEATH_NOTIFICATION⟩ as a ⟨MARKER⟩;
20.   end;
21. if node $n$ receives a "normal" msg from the network then
22.   begin
23.     if node $n$ is in SNAPPING mode then
24.       Save msg to build local state;
25.     Process msg;
26.   end;
```

Figure 2. Adapted Distributed Snapshot Algorithm for WSNs.

Figure 3 presents the pseudo-code of the broadcast algorithm adapted for WSNs.

The adapted broadcast, however, causes a problem. The modified DAS continues working but the adapted broadcast locks when the network splits. The neighbors of the dead sensors know about their deaths, but their own neighbors do not. Those will be waiting for the snapshots to come from the dead nodes indefinitely. For the purpose of having more global states we adopted a fault model that delays the fragmentation of the network.

As an improvement of the collecting step we have implemented the idea of PIF, as we have already explained. When the sink finalizes its own snapshot, it sends a PIF message

```
ALGORITHM Broadcast_WSN:
 1. if Snapshot is finalized then
 2.   Nodes send their local states using ⟨BROADCAST⟩ msgs;
 3. if node $n$ receives a ⟨BROADCAST⟩ msg from node $p$ then
 4.   begin
 5.     if it is the first ⟨BROADCAST⟩ msg received from $p$ then
 6.       Node $n$ sends msg to all its neighbors;
 7.     if node $n$ is the sink node then
 8.       Store local state received in ⟨BROADCAST⟩ msg;
 9.   end;
10. if sink node received all local states then
11.   Build global state;
12. if node $n$ receives ⟨DEATH_NOTIFICATION⟩ msg then
13.   begin
14.     Mark neighbor as dead;
15.     if node $n$ is in BROADCASTING mode then
16.       Treat ⟨DEATH_NOTIFICATION⟩ as a
          ⟨BROADCAST⟩ msg;
17.   end;
```

Figure 3. Adapted Broadcast for WSNs.

to all its neighbors. This message can be anyone and, in this work, we call it a PIF message. The neighbor, that receives the PIF message for the first time, considers the node that has sent it as its parent and forwards a PIF marker to all its neighbors with the exception of its parent. Behaving like this a spanning tree is dynamically built. When the leaves receive PIF messages from all their neighbors, they send their own snapshots to their parents as a feedback for the first PIF message. The other nodes, which are not leaves, will receive the feedback messages with the snapshots from their children. These nodes will join their own snapshots to those of their children and send them to their parents until the sink (root of the spanning tree) has the snapshot of the complete network.

Figure 4 presents the pseudo-code for the adapted PIF algorithm for WSNs.

In case of a network fragmentation, the PIF algorithm still works. It is able to collect the snapshots from the nodes that the spanning tree can still reach. In the PIF algorithm, the number of messages exchanged in the network is smaller than it is in the broadcast algorithm but messages tend to be longer. Both the number and the length of the messages can cause a higher energy consumption. We will evaluate their effects in the next section, where we show the simulation results. One possible solution for the problem of message length is to perform some kind of data fusion as the feedback messages move towards the sink node.

When the network fragments, the nodes on the side of the split network that are not reached by a fixed sink can yet conclude their local snapshots. If we have a mobile sink, these local snapshots could be obtained.

```
ALGORITHM PIF_WSN:
 1. if sink node finalized snapshot then
 2.   begin
 3.     Sink node enters PIF mode;
 4.     Sink node sends ⟨PIF⟩ msg;
 5.   end;
 6. if node $n$ receives ⟨PIF⟩ msg then
 7.   if node $n$ does not finished its local state then
 8.     Store ⟨PIF⟩ msg
 9.   else
10.     if node $n$ is not in PIF mode then
11.       begin
12.         Enter PIF mode;
13.         Parent ← Sender of ⟨PIF⟩ msg;
14.         Decrease # of ⟨PIF⟩ waiting;
15.         Send ⟨PIF⟩ msg to all output channels;
16.       end
17.     else
18.       if ⟨FEEDBACK⟩ msg and is the parent then
19.         begin
20.           Group local states;
21.           Decrease # of ⟨PIF⟩ waiting;
22.         end;
23. if there is no more ⟨PIF⟩ msgs to receive then
24.   begin
25.     Build ⟨PIF⟩ msg will local states;
26.     Send ⟨FEEDBACK⟩ msg to its parent;
27.   end;
28. if node receives ⟨DEATH_NOTIFICATION⟩ msg then
29.   if receiving node is in PIF mode then
30.     if sender is the parent then
31.       Abort PIF in this snapshot
32.     else Treat ⟨DEATH_NOTIFICATION⟩ as a ⟨PIF⟩;
```

Figure 4. Adapted PIF for WSNs.

4. Simulation

The algorithms presented above were simulated, and the metrics used to evaluate them were energy spent, number of exchanged messages and execution time.

4.1. Network simulated

In the simulation we consider that the network is “flat”, i.e., there are no clusters or hierarchy. In this way it is possible to evaluate better the effect of the algorithms on the nodes since all elements in the network take part of the computation in the same way. In hierarchical networks, on the other hand, the cluster leader has a more important role. The sink, the special node where all information captured by the sensors is sent to, is fixed and has more computational resources. It is a very common situation since the energy map tends to be built "outside" of the network. The topology is dynamic, and thus, some nodes can become unavailable ("can die"), what it is also typical in a WSN. The channels are FIFO and reliable (no messages are lost). Furthermore, if a pair of nodes does not have a channel connecting them, it means that one node cannot send or receive messages directly from the other.

We used three topologies to evaluate the algorithms. Each one represents a grid of 30×30 nodes, equally distributed over a geographic region. In the first topology, the sink is connected to only one node at position $(30, 30)$. In this case, all information that passes from the network to the sink or from the sink to the network must pass through this node. If this node dies, the network will be inaccessible by the sink and will become useless. In the second topology, the sink node is connected to n nodes, all of them belonging to side of the grid. Here, many nodes can exchange information with the sink and forward messages to the network. The information coming from the network can reach the sink through different paths and the network will not be inaccessible unless all of them die. The third topology is similar to the first one, excepted that the sink is connected to the node at the center of the grid located at position $(15, 15)$.

We adopt an energy model based on that proposed by [2]. According to it and making some simplifications we have that the energy spent by transmission = $c_1 \times c_2 \times l$, and energy spent by reception = $c_2 \times l$, where l is the message length in bits, $c_1 = 1.4$, an $c_2 = 180$ nJ/bit.

4.2. The simulator

The simulator used in this work is the DAJ simulator [3] that provides a platform to develop distributed algorithms. The execution model consists of a network of nodes, that can be connected by channels and that operate asynchronously and independently of each other. The communication mechanism is based on point-to-point message exchange among connected nodes. The simulator version available does not have libraries for simulating network protocols, which must be developed by the programmer if needed.

4.3. The simulation environment

The simulations were run in a Sun Enterprise 450 server with 2 processors UltraSPARC-II de 250-MHz, 1024 MB of RAM memory, Solaris Operation System version 9, and Java 2 Runtime Environment, Standard Edition 1.4.

The execution time for algorithms DSA_WSN and Broadcast_WSN was about 9 hours using the Topology 1, about 41 hours using the Topology 2, and about 11 hours using the topology 3. The execution time for algorithms DSA_WSN and PIF_WSN was about 30 minutes using the Topologies 1 and 2, and about 20 minutes using the Topology 3.

5. Results

In our simulations we have measured the energy spent, the execution time and the number of messages exchanged considering the DSA algorithm to take the local snapshots, and the broadcast and PIF algorithms to collect all snapshots. We have measured these in the three topologies described earlier.

In this work we are not considering the computation of any specific application. Thus, it is expected that the energy spent will be directly related to the number of messages exchanged among the sensor nodes. The number of messages sent by each node is constant in all simulated algorithms. The messages that are determinant to the format of the energy map are the messages received by the nodes.

In the case of using the broadcast algorithm to collect the local states, the number of messages sent by the algorithms (DSA and broadcast) was constant and equal to 902 for all nodes in the three mentioned topologies, considering a network with 900 nodes. In the DSA algorithm step, each node needs to send only a message, at the instant of propagating the marker message to their neighbors. In the broadcast step, each node sends 901 messages to propagate the local snapshots, that corresponds to the snapshots of the 900 nodes in the network plus the sink message, that participates in the broadcast.

The most favorable topology in the case of broadcast was Topology 1. In this case the energy consumption of the network was between 35 mJ and 45 mJ. The nodes have received between 1800 and 2700 messages. In the Topology 3, almost all network spent about 45 mJ. Almost all nodes received about 2800 messages. Topology 2 was the most energy consumer, where most nodes spent about 55 mJ. The number of messages received by most of the nodes were almost 3600 messages, which it is 1800 messages more than the number of messages received by the most expensive nodes on the other topologies. This is because the sink node is more connected than in the other topologies.

Figure 6 presents the energy consumption and the messages received, for the three topologies, using now PIF as the algorithm for collecting local states. In this case, the number of messages sent was constant and equal to 3 for any node on the three topologies, notwithstanding the network size. As we have shown, on the DSA step each node needs to send only a message. On the PIF step, each node needs to send two messages, one to its neighbors with the message to initialize the PIF algorithm and another to its parent with its own snapshot plus the its children's snapshots.

The maps shown in the first column of Figure 6 represent the energy consumed (mJ) at each node in the network after the first snapshot considering both the DSA and PIF algorithms. The maps in the second column represent the number of messages received at each node in the network. The maps of both energy consumption and received messages are very similar, as expected. As mentioned before, in our simulation we are not considering any specific application. Consequently, the energy spent depends basically on the number of messages exchanged among the nodes. In this case, the map format is mainly related to the format of the spanning tree.

In the case of PIF, the distribution of the energy spent was likewise the spanning tree built. The nodes near the spanning tree root (near the sink) spent more energy and the leaves (the nodes more distant from the sink) spent less energy. The most favorable topology in the case of PIF was Topology 1, where most nodes spent between 0.12 and 0.14 mJ, increasing in the sink direction. The messages tend to become larger as they get closer to the sink because there is no data compression or fusion, only grouping. Looking to the map of messages received, we can observe that the number of messages also increases in that direction. It can be justified by the format of the spanning tree generated by the PIF algorithm. Moreover, when a node sends a message to its parent, all nodes reached by this node also receive the message. In spite of ignoring messages that are not addressed to them, these nodes still spend energy by hearing these messages. This is a point that can be improved.

The average number of messages received in the case of PIF was between 5 and 10 messages. In the PIF case, the boundary still presents fewer messages received and low energy spent. It can be once again justified by the lower connectivity among these nodes and the network and the format of the spanning tree.

In all topologies, most nodes spent about 0.12 mJ. In Topology 3 there was a higher fluctuation in energy spent among the nodes. In this topology the nodes directed to the sink spent much more energy than in the other topologies.

We can observe that in the simulations where the PIF algorithm was used to collect the local states, the energy consumption was on average three magnitudes less than when the broadcast algorithm was used. Messages in the PIF algorithm tend to be longer than that in the broadcast algorithm. Despite this, the PIF algorithm yet outperformed the broadcast algorithm in terms of energy spent, that reflects the number of messages exchanged in these two algorithms. In the case of PIF, the number of messages exchanged by most nodes was less ten messages, and in the case of broadcast was between 2700 and 2900 messages in the most favorable topology.

6. Conclusions

A WSN represents an important class of network that can be used in a variety of situations. Traditional distributed algorithms cannot be directly applied to these networks because of the distinguishing characteristics of WSNs. Thus, it is necessary to design new versions of these algorithms. This work presented how the DSA algorithm can be adapted for WSNs. Furthermore, we have shown how this algorithm can be used to obtain a very important information in WSNs that is the energy map. From this map, we can have a better idea of the remaining energy available in the network. We can define, for example, where the sink must be placed and which algorithm for obtaining the local state must be used to decrease the number of messages exchanged in the network, increasing its life time.

The DSA algorithm, together with PIF, is a very efficiency solution in terms of messages exchanged, energy spent and execution time when compared to the broadcast algorithm. Yet, many improvements can be introduced in respect to the use of these algorithms. When, in the PIF, the children nodes send messages to their parents, all nodes reached by them, receive these messages, wasting energy. The nodes can, for example, turn off their radios when receiving messages not addressed to them.

Another improvement to the PIF algorithm, would be a parent node wait for the feedback from a fraction of its children or for a period of time. The latter could solve the problem of unavailable nodes. We are currently working in these propositions.

References

[1] K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, February 1985.

[2] A. P. Manish Bhardwaj, Timothy Garnett. Upper bounds on the lifetime of sensor networks. Technical report, Massachusetts Institute of Technology, Cambridge, MA 02139, March 2001.

[3] W. Schreiner. A java toolkit for teaching distributed algorithms. In *Proc. of the 7th ACM Annual Conf. on Innovation and Technologoy in Computer Science Education*, 2002. http://www.risc.uni-linz.ac.at/people/schreine/.

[4] A. Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, 29:23–35, 1983.

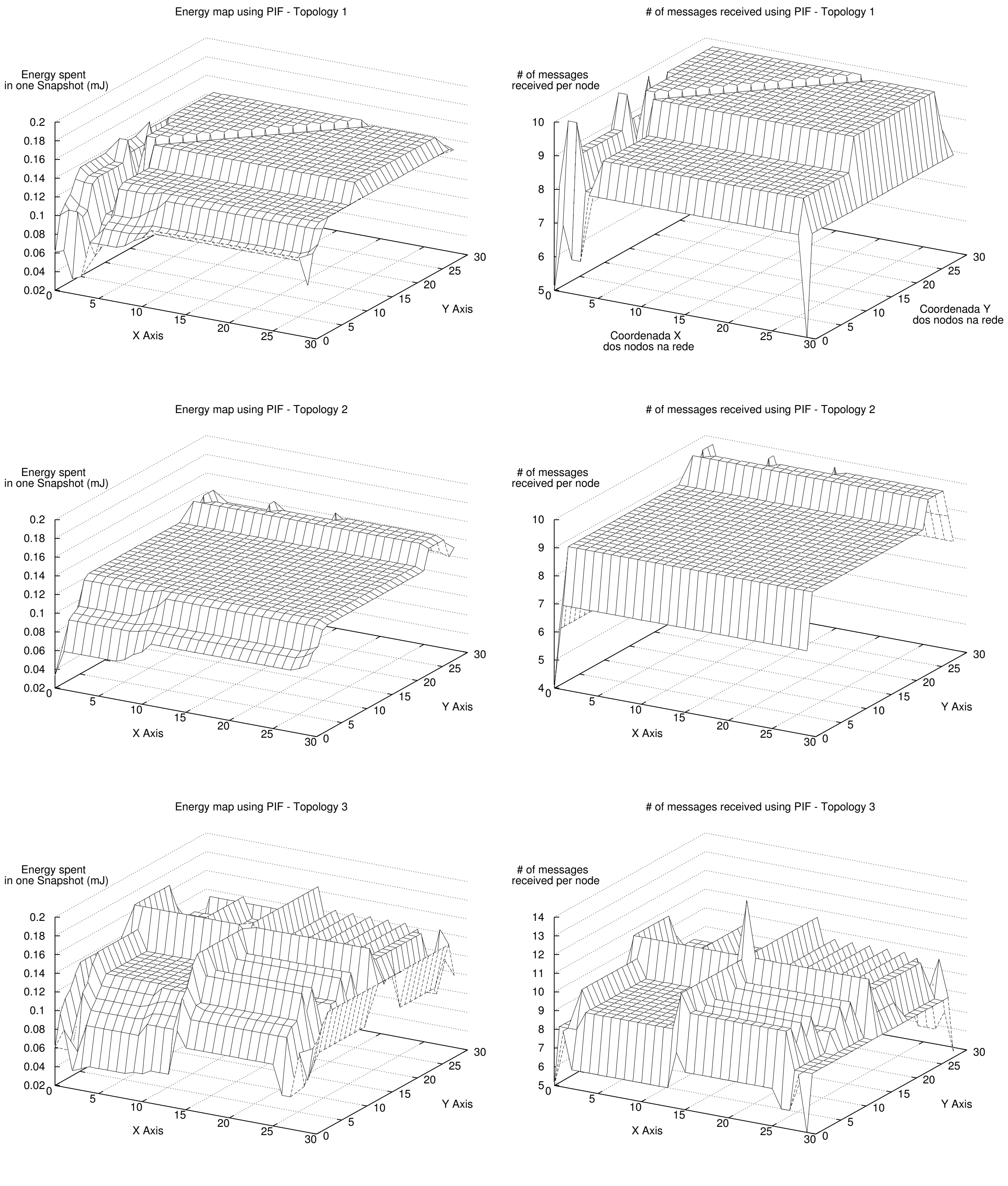

Figure 5. Energy consumption and messages received using PIF