Formal Modeling and Validation of a Power-Efficient Grouping Protocol for WSNs [☆]

Fatemeh Kazemeyni[a,b,*], Einar Broch Johnsen[a], Olaf Owe[a], Ilangko Balasingham[b]

[a]*Department of Informatics, University of Oslo, Norway*
[b]*The Intervention Center, Oslo University Hospital, Norway*

Abstract

Wireless sensor networks consist of resource-constrained nodes; especially with respect to power resources. Often, the replacement of a dead node is difficult and costly; e.g. a node may be implanted in the human body. Therefore, it is important to reduce the total power consumption of WSNs. The major consumer of power is the data transmission process. This paper considers nodes which cooperate in data transmission in terms of a *group*. A mobile node may move to a new location, in which it is desirable for the node to join a group. We propose a protocol to allow nodes to choose the best group in their signal range, using coalitional game theory to determine what is beneficial in terms of power consumption. The protocol is formalized as an SOS-style transition system. This formalization forms the basis for an implementation in the rewriting logic tool Maude, so the protocol can be validated using Maude's model exploration facilities. First, we prove the correctness of our proposed protocol, by searching for failures through all possible behaviors for given initial states. For these searches, the grouping is done correctly in all reachable final states of the model. Second, we simulate the model behavior to quantitatively analyze the efficiency of the proposed protocol. The results show significant improvements in power efficiency.

[☆]This work was done in the context of the project *Connect* funded by the Norwegian Research Council and the EU project FP7-231620 *HATS*.

[*]Corresponding author.

Email addresses: `fatemehk@ifi.uio.no` (Fatemeh Kazemeyni), `einarj@ifi.uio.no` (Einar Broch Johnsen), `olaf@ifi.uio.no` (Olaf Owe), `ilangkob@medisin.uio.no` (Ilangko Balasingham)

Keywords: wireless sensor networks, game theory, grouping, formal modeling

1. Introduction

A wireless sensor network (WSN) typically consists of sensor nodes with sensing, computing, and communication devices. The main goal of the WSN is to gather data from the environment and transmit it to a sink node. WSNs are usually self-configured ad-hoc networks with mobile nodes.

The physical size of sensor nodes is very small, which introduces challenges for the design and management of WSNs. Especially, restrictions in power resources need to be considered in order to improve the longevity of the nodes. Data transmission is expensive, therefore, the management of communication between nodes is an important factor in power efficiency of the network. *Cooperation* between sensor nodes can potentially reduce the total power consumed for data transmission in the whole network by replacing multi-hop with traditional single-hop communication [1].

Grouping is a method to organize node cooperation in a WSN [2]. A group has a selected node called the *leader* which is responsible for receiving data from the group members and for the communication with the outside of the group. Inside a group, nodes help each other to transmit data to the leader using multi-hop instead of single-hop communication, thereby expecting to reduce the consumed power.

Nodes which are close to each other may in principle communicate using less power. By cooperating inside a group, the group's members can decrease their transmission power to minimum and still reach the leader. However, if nodes do not have fixed locations, the network topology can change. Nodes should compute the most efficient way to communicate in the network. Consequently, the group structure of the network may need to evolve. In a self-organizing network with a dynamic topology, a node which moves may want to join a group to have a cheaper communication and the group needs to decide whether to accept the node.

This paper proposes a new protocol to decide which group could be the best for the node to join. The node chooses a group such that joining it is beneficial for the node, for the group, and also whole the network. In order to decentralize the grouping process, in the proposed protocol the nodes choose the best group to join with respect to the *total energy* of the network.

Our grouping protocol needs an underlying routing protocol which finds the cheapest route between a node and a group leader with respect to power consumption. For this purpose, we propose a *power-sensitive AODV* routing protocol to find the cheapest routing path between group members. Based on the power-sensitive AODV, our grouping protocol applies coalitional game theory to decide on the best group membership.

In this paper, we combine the power-sensitive AODV protocol proposed in [3] with an extension of the grouping protocol originally proposed in [4]. This extension allows the nodes to choose the best group among different possible groups, in contrast to the protocol in [3] where the leader is responsible for deciding about the node's membership in the group, solely based on the local benefit of the group. Compared to the previous papers [3, 4], we use the framework of *structural operational semantics* (SOS) [5] to formalize our proposed protocol. This way a more abstract representation of the protocol is obtained, compared to our earlier work.

Protocol analysis is traditionally based on simulation tools. Instead, we use formal techniques to analyze our protocol. Formal techniques not only provide a more abstract model, but make it possible to prove protocol correctness in addition to simulate its behavior. Simulation-based tools can provide useful statistical results from protocol's behavior, but since it is practically impossible to test exhaustively all the behaviors of networks, these tools can not prove the correctness of a protocol. By using formal techniques, we can inspect all reachable states of a system and prove its correctness. Although the abstract formalization of the proposed protocol is given in an SOS style, the formalization can easily be transformed into a model in rewriting logic [6]. Consequently, the protocol can be analyzed using the rewriting logic tool Maude [7]. We show correctness of the proposed protocol.

1.1. Related Work

Approaches to energy conservation for WSNs that have been proposed in the literature could be categorized in three: duty cycling, data-driven, and mobility-based approaches [8]. The *duty cycling* approach is concerned with networking subsystems and sleep/wake-up scheduling algorithms. These methods, such as in [9, 10, 11, 12], try to identify efficient subsets in the network and schedule the activity of network nodes. The purpose of the *data-driven* approach is to reduce the amount of data that is transmitted between sensors or the sink node, and considers data compression methods [13, 14, 15].

The *mobility-based* method such as presented in [16, 17, 18, 19], can be categorized as mobile-sink and mobile-relay methods, depending on the type of the mobile entity. Mobile entities can gather the data from the nodes by using short range communication, which is an efficient way of communication with respect to energy. A similar idea of grouping the nodes for power efficiency is used in [20, 21]. The proposed grouping technique fits best in the duty cycling approach, because the group members and the group leader arrange their duties in order to cooperate with each other, and thereby conserve the total energy of the group and the network. Our algorithm manages the duties of the nodes in the network. We do not specifically use sleep/wake-up algorithms to schedule the duties, but instead use the groups in the network and assign different duties to different group's roles.

Game theory is categorized in noncooperational and cooperational (coalitional) games. Noncooperational game theory has been used to reduce the power consumption of sensor nodes, applying a utility function to find the Nash equilibrium [22, 23, 24]. Coalitional game theory is applied to reduce the power consumption in WSNs by [25], proposing a merge and split approach for coalition formation. The authors calculate the value of the utility function for every possible permutation of nodes and find groups with the best utility value. This is as far as we know the only previous work that uses coalitional game theory for grouping the sensor networks. In contrast we develop and formalize a protocol which considers nodes which may need to join a new group, without reorganizing the entire WSN.

WSNs present interesting challenges for formal methods, due to their resource restrictions and radio communication. This has led to research on how to develop modeling languages or extensions which faithfully capture typical features of sensors; e.g., mobility, location, radio communication, message collisions. In addition, WSNs need communication protocols which take resource usage into account. There is a very active field of research on protocol design for WSNs. However, protocol validation is mostly done with simulation-based tools, using NS-2, OMNeT+, and extensions such as Castalia [26] and SensorSim [27]. In contrast to these tools, our approach is based on formal modeling which allows to have systematic and comprehensive investigation of model properties and correctness.

Formal techniques are much less explored in the development and analysis of WSNs, but start to appear. Among automata-based techniques, the TinyOS operating system has been modeled as a hybrid automaton [28] and UPPAAL has been applied to the LMAC protocol [29] and to the temporal configuration parameters of radio communication [30]. The CaVi tool combines simulation in Castalia with probabilistic model checking [31].

A recent process algebra for active sensor processes includes primitives for, e.g., sensing [32]. The CREOL object-oriented modeling framework has an extension for heterogeneous environments, which includes radio communication [33]. The Temporal Logic of Actions has been used for routing tree diffusion protocols [34]. Maude is an expressive, high-level, reflective language with a formal semantics based on rewrite logic. It supports a wide range of applications efficiently, using specifications in terms of equations describing functional data types, and rewrite rules, describing state changes. Maude can also support meta-programming, object orientation, real time, and probabilistic modeling. It provides an executable environment for modeling, testing, and validating different applications [35, 36, 7]. Ölveczky and Thorvaldsen show how a rich specification language like Maude is well-suited to model WSNs, using Real-Time Maude to analyze the performance of the OGCD protocol [37]. They demonstrate that Maude can be used to detect flaws early in high-level designs. In addition, it can be applied to develop executable prototypes and to generate code, very quickly. It also has a wide range of analysis tools.It is possible to debug formal partial or incomplete specifications and also do exhaustive model-checking, or bounded model exploration. For critical systems, full formal verification using theorem proving tools are provided. Their approach has been combined with probabilistic model-checking to analyze the LMST protocol [38]. We follow this line of research and use Maude as a tool to develop a grouping protocol [2] for WSNs. Compared to other formal languages Maude allows infinite state systems and variables over types with infinite value sets. For such systems bounded breadth first search and statistical model checking are valuable.

1.2. Paper Overview

Section 2 introduces WSNs and grouping, and Section 3 coalitional game theory. Section 4 explains the power-sensitive extension to the AODV routing algorithm. Section 5 the proposed group membership protocol based on coalitional game theory. Section 6 presents the formalization of the protocol in an SOS style. Section 7 discusses the Maude implementation and the analysis of the model and Section 8 concludes the paper.

2. Grouping the Sensor Nodes

A sensor network is typically a wireless ad-hoc network, in which the sensor nodes support a multi-hop routing algorithm. In these networks, communication between nodes is generally performed by direct connection (single-hop) or through multiple hop relays (multi-hop). Multi-hop ad-hoc wireless networks use more than one wireless hop to transmit information from a source to a destination.

Wireless sensor nodes use routing protocols to communicate with each other and to find the path to transmit the data that is sensed from the environment to the designated sink node (or nodes). In most of these protocols, the nodes broadcast their data to all nodes that are within the range of their data transmission. This range is determined by the power used for transmission. In general, sensor nodes use their maximum data transmission power to cover a larger area and reach more nodes, both for data transmission and for routing. For example, in the standard AODV protocol [39], a node that moves or enters the network broadcasts a routing request package (*Hello*) with maximum power to find neighbors. Due to node mobility, a node may need a new routing path, so it rebroadcasts a routing request to its neighbors using maximum power.

When a large number of sensor nodes are placed in the environment, neighbor nodes may be very close to each other. In this case, the transmission power level for communication with a neighbor can be kept low. Since nodes can cooperate with each other to transmit data, multi-hop communication in sensor networks is expected to consume less power than the traditional single-hop communication [1]. Furthermore, multi-hop communication can effectively overcome some signal propagation effects experienced in long-distance wireless communication.

Grouping. Grouping is a method for cooperation between nodes, in which nodes belong to distinct groups [2]. When nodes form a group, they help each other to transfer data in a more organized way. Each group has a group *leader*; i.e., a designated member which receives data from the group members and communicates with other group leaders in order to route the data to its destination. Inside the group, it is not always necessary for a node to use its maximum transmission power. Instead, the group members can decrease the power consumed for communication and use their minimum transmission power to reach the group leader. The leader should be chosen carefully to have enough energy to do its responsibilities. In this paper we do not focus on the leader selection issue but rather on the group management. We assume that Leaders are fixed nodes with renewable energy resource.

Sensor nodes in the real world are not designed to directly support grouping. We therefore assume that nodes know nothing about the grouping process. In contrast, the group leaders are special nodes that process the information of the newly entered sensor nodes and decide about their possible group membership. Group formation could be done by using different techniques. In general, the grouping of nodes can be done based on particular characteristics or distance. In the first case, a correlation among the sensors could be found by using vector quantization [40]. For example, all sensor nodes that have similar sensed data could be placed in one group. In the second case, the sensor nodes form different groups based on the distance. With this technique, a node's location is the important factor for group formation, but to have a better grouping, other factors such as interference could also be considered for group formation. The location of the nodes can be determined using different methods, such as GPS.

3. Coalitional Game Theory

Game theory [41] can be used to analyze behavior in decentralized and self-organizing networks. Game theory typically sees the nodes as players and models the choice of strategies of self-interested players, in order to capture the interaction of players in an environment such as a communication network. A game consists of

- a set of *players* $\mathbf{N} = \{1, 2, ..., n\}$;
- an indexed set of *possible actions* $A = A_1 \times A_2 \times ... \times A_n$, where A_i is the set of actions of player i (for $0 < i \leq n$);
- a set of *utility functions*, one for each player. The utility function u assigns a numerical value to the elements of the action set A; for actions $x, y \in A$ if $u(x) \geq u(y)$ then x must be at least as preferred as y.

Game theory can be either *cooperative* [41] or *noncooperative* [42]. Noncooperative game theory studies the interaction between competing players, where each player chooses its strategy independently and the goal of each player is to improve its utility or reduce its cost [25]. In cooperative games, groups of players are formed, called coalitions. The players try to find a coalition to strengthen their position in the game and make an agreement to act as a simple entity. Coalitional games have proved useful to design fair, robust, and efficient cooperation strategies in communication networks. In a coalitional game (N, v) with N players, the coalition value or utility of a coalition is determined by a *characteristic function* $v : 2^N \to \mathbb{R}$ which applies to coalitions of players. Most coalitional games have *transferable utility* (TU); i.e., the utility of a coalition can be distributed between the coalition members according to some notion of fairness. However, for some scenarios a coalition's utility cannot be captured by a single real value, or rigid restrictions are needed on the distribution of the utility. These games are known as coalitional games with *nontransferable utility* (NTU). In an NTU game, the payoff for each player in a coalition S depends on the actions selected by the players in S. The core of the coalitional game (N, v) is the set of payoff allocations that guarantees that no player has an incentive to leave N to form another coalition [25].

4. Power-sensitive AODV Routing protocol

We propose a *power-sensitive Ad-hoc On-Demand Distance Vector (AODV)* routing protocol, based on AODV protocol [39], which is reactive; i.e., routes are created at need. It uses traditional routing tables, one entry per destination, and sequence numbers to decide if the routing information is up-to-date and to avoid loops. Note that *power-sensitive AODV* maintains time-based states in each node; if a routing entry has not been used recently, it expires and the node's neighbors are notified. Route discovery is based on query and reply cycles, and route information is stored in all nodes along the route as route table entries. The *power-sensitive AODV* protocol works as follows:

1. Nodes broadcast *Hello* messages to detect and monitor links to neighbors.
2. The route discovery process starts when a node which requires a route to another node, broadcasts a *rreq* message.
3. If the neighbor which receives this message has no route entry for the destination, or this entry is not up-to-date, the *rreq* is re-broadcasted with an incremented transmission power which shows the consumed power in the path.
4. While the *rreq* message is broadcasting through the network, each node remembers the reverse path to the source node.

5. If the receiver node is the destination or it has a routing path to the destination with a sequence number larger or equal to that of *rreq*, a *rrep* message is sent back to the source. The route to the destination is established when a *rrep* message is received by the original source node.
6. A source node may receive multiple *rrep* messages with different routes. It then updates its routing entries if the information is new in the sense that the *rrep* has a greater sequence number.

Note that the AODV routing protocol counts the number of hops (the length of the path) and finds the shortest path, while *power-sensitive AODV* finds the cheapest path regarding to the consuming power of the nodes in the path.

5. A Protocol for Deciding Group Membership

Consider the grouping problem for wireless sensor networks as a coalitional game. The sensor nodes are the players and the game is concerned with whether a node should join a group or not. The goal is to reduce the total power consumption in the network, so we need a utility function which reflects the power consumed for data transmission and signal interference. The utility function proposed by Goodman *et al.* [43] appears to be a suitable choice when power consumption is an important factor of the model [44]:

$$w(power_j, \delta_j) = (\frac{R}{power_j})(1 - e^{-0.5\delta_j})^L. \tag{1}$$

When applying w to a node j, $power_j$ is the power used for message transfer by j and δ_j is the signal to interference and noise ratio (SINR) for j. In addition, R is the rate of information transmission in L bit packets in the WSN.

Nodes can transfer data with different amounts of power. Let $power^{max}$ denote the maximum transmission power and $power_j^{min}$ the minimum transmission power for each node j, such that $0 \leq power_j^{min} \leq power_j^{max}$. When a node j cooperates in a group, it uses $power_j^{min}$ for message transmission, and otherwise $power_j^{max}$. Consider a network of nodes $N = \{1, \ldots, n\}$. If all the nodes in N cooperate, we have:

$$\sum_{j=1}^{n} w(power_j, \delta) = \sum_{j=1}^{n} w(power_j^{min}, \delta)$$

Without cooperation $power_j^{max}$ is assigned to $power_j$. Observe that if this utility function were applied naively, it would always be beneficial for nodes to form a coalition, as the result of decision making is the same for every topology of the network and every group.

However, in reality all the cooperating nodes use power in order to transmit data to the group leader, so it is not sufficient to only consider the power consumption of the original sender of data in the utility function. Although each node uses its minimum power to transmit data, the node's total power usage depends on the number of messages it needs to transmit. Each node on the route between the source node and the leader, needs to send its own data as well as the data that it has received from the previous node. In general, the power consumption for the intermediate nodes will increase. We modify the utility function (Formula 1) to capture the overall power usage needed to transmit the data from the node to the leader following a given path:

$$u(power_j, \delta_j) = (\frac{R}{\sum_{n \in RP_{j,Leader}} power_n^{min}})(1 - e^{-0.5\delta_j})^L, \tag{2}$$

where the set $RP_{j,Leader}$ contains all nodes in the routing path between node j and the leader. This utility function is similar to Formula 1 except that the power that is applied is the sum of the power consumed by all the nodes in the routing path through which data is transmitted from the sender to the leader.

The power consumed for routing data from a non-member to the leader follows Goodman *et al.* (Formula 1) and is based on maximum power single hop:

$$w(j, \delta_j) = (\frac{R}{power_j^{max}})(1 - e^{-0.5\delta_j})^L$$

Using the utility function u, the leader can decide about the membership of a new node with more realistic estimations. The result depends on the specific topology, so coalition is not always beneficial. Consequently, it is more beneficial for the node to follow a path through the group than to act individually when $w(j, \delta_j) < u(j, \delta_j)$ holds. To calculate the power that is used in the cooperation, we have proposed a *power-sensitive AODV* routing protocol, modifying AODV to find the cheapest path between the node and the leader in terms of power (more details in the section 4). The leader may then decide to add the node to its group and sends an *Invitation* message to the node. A node may receive several *Invitation* messages from different leaders due to the intersection of groups. In this paper, we consider how the node may select the best group to join, using game theory. It chooses to join the group which is the most beneficial for the overall network.

Consider a group i with leader *Leader*. Let M be the set of nodes which can reach *Leader* with $power^{max}$ and N the set of group members. Let the accumulated *group utility value* g_i be determined by the sum of the utility values for communication with *Leader*:

$$g_i(M, N) = \sum_{j \in M} w(j, \delta_j) + \sum_{j \in N} u(j, \delta_j)$$

The group membership protocol extends the power-sensitive AODV protocol as follows:

1. Node j sends a *Hello* message with maximum power to all group leaders within range;
2. Each group leader runs the power-sensitive AODV protocol to find the cheapest path for j as a potential group member and evaluates the benefit of group membership for j: $g_i(M \cup \{j\}, N) < g_i(M, N \cup \{j\})$;
3. If membership is beneficial, group i's leader sends an *Invitation* message to j, including the utility values $v^i_{\text{old}} = g_i(M \cup \{j\}, N)$ and $v^i_{\text{new}} = g_i(M, N \cup \{j\})$;
4. Node j may receive many *Invitation* messages, which are processed sequentially. By assumption, j is currently in group a and knows v^a_{old} and v^a_{new}; For each invitation, j computes $v^i_{\text{new}} - v^i_{\text{old}} > v^a_{\text{new}} - v^a_{\text{old}}$. If this is the case, j accepts the invitation from i and sends a *Leave* message to a with the value $v^a_{\text{new}} - v^a_{\text{old}}$;
5. The *Leader* receives an *Accept* message from j and updates its utility approximation;
6. The *Leader* receives a *Leave* message with value v_j, and updates its utility approximation v to $v - v_j$.

The core of this game is not empty and nodes can form groups. Assuming two disjoint groups i and a with $v(a \cup i) = 0$, the core is empty [25]. In our game $i \cap a \neq \emptyset$, $g_i(M, N) = v(M, N)$, and $g_i(M \cup \{j\}, N) < g_i(M, N \cup \{j\})$, which proves that the core is not empty and that no node or group of nodes has incentive to leave the coalition.

6. The Formal Model of the Proposed Protocol

In this section, we define a formal model of the group membership protocol as a transition system in an SOS style [5]. Transition rules apply to system configurations. We explain the rules and auxiliary functions modeling wireless message passing, node movement, and the routing protocol, as well as the evaluation of the utility function. The rules are presented using the following SOS notation:

$$\frac{\textsc{(Rule Name)}\quad \textit{Conditions}}{\textit{conf} \longrightarrow \textit{conf}'}$$

Here the pattern *conf* locally transforms a configuration to *conf*′ if the *Conditions* hold, given a match between *conf* and the configuration. In order to distinguish trivial conditions from non-trivial ones, we will use the notation $x := e$ when the variable x is introduced as a short-hand for the expression e in the rule, similar to a let-construct. Mathematically $x := e$ can be understood as $x = e$.

Our rules involve one node object and at most one incoming message, as typical for distributed asynchronous system, and each rule has a conclusion of form

$$\textit{msgs node conf} \longrightarrow \textit{node}' \textit{ newmsgs conf}$$

where *msgs* are the messages consumed by the node, *newmsgs* are the messages produced by the node, and *node*′ is the node in its resulting state. Without the *conf* variable this would represent local activity of each object, and concurrent execution could be represented by standard SOS context and concurrency rules. However, we are modeling wireless broadcast communication and must dynamically determine which nodes are within the transition range of a sender. Because the exact subset cannot be determined locally, we include the entire configuration in the rules by means of the variable *conf*.

A *system configuration* is a multiset of objects and messages, where the node objects are assumed to have unique identifiers. A node has the form of $\mathbf{node}(o, p, e, n, l)$, where

- o is the identifier of the node,
- p is the position of the node in the form of (x, y),
- e is a pair of $(\textit{power}, \textit{energy})$ where *power* is the power capability of the node and *energy* is the total remaining energy in the node,

- n is a tuple $(\mathit{routing}, \mathit{utility}, \mathit{neighbor}, \mathit{reqID}, \mathit{prev})$ where,
  - *routing* is the routing table of the node,
  - *utility* is a tuple of (u^n, u^o) where are the new and old utility value of the node's group,
  - *neighbor* is the list of node's neighbors
  - *reqID* is the sequence number of the last received request message,
  - *prev* is the last node in the routing path,
- l is the identifier of the node's leader.

Figure 1 defines observer functions used to extract a node and its different attributes from a given configuration. Messages have the general form of $\mathbf{msg}(h, s, d)$, where

- h is the header of the message which is the form of $\mathit{name}(c)$, where *name* indicates the message name and c is the content of the special message,
- s is the message's source node, and
- d is the identifier of the message's destination node.

6.1. Unicast, Multicast, and Broadcast

We consider three types of message passing in WSNs: unicasting, multicasting, and broadcasting. We use auxiliary functions to capture message passing in our model. These functions are defined in Figure 2 and explained below. In WSNs, messages are sent through the atmosphere and can only

$\mathit{Node}(o, \mathit{conf}) = \mathbf{node}(o, p, e, n, l)$	if $\mathbf{node}(o, p, e, n, l) \in \mathit{conf}$
$\mathit{Position}(o, \mathit{conf}) = p$	if $\mathit{Node}(o, \mathit{conf}) = \mathbf{node}(o, p, e, n, l)$
$E(o, \mathit{conf}) = e$	if $\mathit{Node}(o, \mathit{conf}) = \mathbf{node}(o, p, e, n, l)$
$N(o, \mathit{conf}) = n$	if $\mathit{Node}(o, \mathit{conf}) = \mathbf{node}(o, p, e, n, l)$
$\mathit{Power}(o, \mathit{conf}) = \mathit{power}$	if $E(o, \mathit{conf}) = (\mathit{power}, \mathit{energy})$
$\mathit{Energy}(o, \mathit{conf}) = \mathit{energy}$	if $E(o, \mathit{conf}) = (\mathit{power}, \mathit{energy})$
$\mathit{Routing}(o, \mathit{conf}) = r$	if $N(o, \mathit{conf}) = (r, u, n, \mathit{id}, p)$
$\mathit{Utility}(o, \mathit{conf}) = u$	if $N(o, \mathit{conf}) = (r, u, n, \mathit{id}, p)$
$\mathit{Neighbor}(o, \mathit{conf}) = n$	if $N(o, \mathit{conf}) = (r, u, n, \mathit{id}, p)$
$\mathit{RequestID}(o, \mathit{conf}) = \mathit{id}$	if $N(o, \mathit{conf}) = (r, u, n, \mathit{id}, p)$
$\mathit{Prev}(o, \mathit{conf}) = p$	if $N(o, \mathit{conf}) = (r, u, n, \mathit{id}, p)$
$u^n = \mathit{new}$ and $u^o = \mathit{old}$	if $\mathit{Utility}(o, \mathit{conf}) = (\mathit{new}, \mathit{old})$

Figure 1: Observer functions on a node o in a configuration conf.

be seen by the nodes which are in the sender's signal range. This is captured by the function *inrange* which determines whether the two nodes s and d are within transmission range given their positions p_s and p_d. We use $\mathbf{distance}(p_s, p_d)$ to find the distance between two nodes s and d. In addition, $\mathbf{reach}(\mathit{power})$ is used in order to define the distance that a node's signals can be received by using transmission power *power*.

```
function $\mathit{inrange} : (Nat \times Nat) \times (Nat \times Nat) \times Nat \longrightarrow \mathit{Bool}$
$\mathit{inrange}(p_s, p_d, \mathit{power}) = \mathbf{distance}(p_s, p_d) < \mathbf{reach}(\mathit{power})$

function $\mathit{unicast} : Header \times Id \times Id \times \mathit{Configuration} \times Nat \longrightarrow \mathit{Configuration}$
$\mathit{unicast}(\mathit{name}(c), s, d, \mathit{conf}, \mathit{power}) =$
   if not $\mathit{inrange}(Position(s, \mathit{conf}), Position(d, \mathit{conf}), \mathit{power})$
   then $\mathit{none}$ else $(\mathbf{msg}(\mathit{name}(c), s, d))$

function $\mathit{multicast} : Header \times Id \times Id \times \mathit{Configuration} \times Nat \longrightarrow \mathit{Configuration}$
$\mathit{multicast}(\mathit{name}(c), o, \emptyset, \mathit{conf}, \mathit{power}) = \mathit{none}$
$\mathit{multicast}(\mathit{name}(c), o, \{o'\} \cup os, \mathit{conf}, \mathit{power}) =$
   $\mathit{unicast}(\mathit{name}(c), o, o', \mathit{conf}, \mathit{power})\ \mathit{multicast}(\mathit{name}(c), o, os, \mathit{conf}, \mathit{power})$

function $\mathit{nodes} : \mathit{Configuration} \longrightarrow \mathit{OidSet}$
$\mathit{nodes}(\mathit{none}) = \emptyset$
$\mathit{nodes}(\mathit{message}\ \mathit{conf}) = \mathit{nodes}(\mathit{conf})$
$\mathit{nodes}(\mathbf{node}(o, e, p, n, l)\ \mathit{conf}) = \mathit{nodes}(\mathit{conf}) \cup \{o\}$
$\mathit{nodes}(\mathit{conf}) = \emptyset$ [otherwise]

function $\mathit{broadcast} : Header \times Id \times \mathit{Configuration} \times Nat \longrightarrow \mathit{Configuration}$
$\mathit{broadcast}(\mathit{name}(c), o, \mathit{conf}, \mathit{power}) = \mathit{multicast}(\mathit{name}(c), o, \mathit{nodes}(\mathit{conf}) \setminus \{o\}, \mathit{conf}, \mathit{power})$

function $\mathit{findPower} : \mathit{List}[Nat \times Nat \times Nat] \times Nat \longrightarrow Nat$
$\mathit{findPower}(\mathit{none}, x) = 0$
$\mathit{findPower}((d, o, p)\ nl, x) =$ if $x = d$ then $p$ else $\mathit{findPower}(nl, x)$
```

Figure 2: Auxiliary functions for message passing in WSNs. Equations marked with **otherwise** apply when no other equation match.

Unicast is modeled by a function *unicast* which takes a message header, a sender id, a destination id, a configuration and the transmission power of the message. The function returns a configuration which is empty if the sender and destination nodes are not within their transmission range in the configuration. Otherwise, the function returns a message. Note that this equation removes a message which cannot reach its destination, depending on the positions of the nodes at sending time.

Multicast is modeled by a function *multicast* which is similar to unicast, but with a set of destination identifiers and performs unicast to all destination nodes in the set. Here, *os* denotes a set of object identities.

Wireless broadcasting does not have any particular destination, and the message is sent to all the nodes in the transmission range. In order to model the broadcast, we need to have access to all the nodes in the configuration. The multiset of all nodes in the configuration *conf* is given by the function $nodes(conf)$. Broadcast is modeled by a function *broadcast* which takes a message header, a sender id but no specific destination id, a configuration and the transmission power of the message. The function returns a set of messages to all the node within the transmission range of the sender.

Function *findPower* retrieves the power that is needed to route the data to a special destination, from the routing table. The routing table is a list of triples, including the destination node ID, the next node in the routing path to the destination, and the required transmission power.

6.2. Node Movements

In most WSNs, nodes can move and change their location. Therefore, a WSN model should provide suitable rules for changing the position of the nodes. In our model, nodes can move freely within the area that is bounded by predefined borders. The node movements are captured by a rule RANDOM MOVING, given in Figure 3, which non-deterministically changes the location of the node inside of the borders of the environment. Here, $insideBorders(p')$ checks if p' is a position inside the predefined borders of the environment, and the message *Moved* is a message from the node to itself to indicate that the position is changed and that the grouping process should be restarted. In the time of moving, the node also broadcast an *UpdateRT* message to all its neighbors, to inform its movement. The neighbors which receive this message, remove the entries in their routing table, which includes the moving node. The UPDATE ROUTING TABLE rule captures this process. In this rule $RetriveEntries(o, routing)$ finds all the entries in the routing table *routing* that include node o.

(Random Moving)

$$\frac{\begin{array}{c} p \neq p' \\ insideBorders(p') \\ e' := (power, energy - power^{max}) \\ m := broadcast(UpdateRT, o, conf, power^{min}) \end{array}}{\mathbf{node}(o, p, e, n, l)\ conf \longrightarrow\ \mathbf{node}(o, p', e', n, l)\ \mathbf{msg}(Moved, o, o)\ m\ conf}$$

(Update Routing Table)

$$\frac{\begin{array}{c} entries := RetriveEntries(o_1, Routing(o, conf)) \\ e' := (power, energy - power^{min}) \\ n' := n[routing \Rightarrow routing \setminus entries)] \end{array}}{\mathbf{msg}(UpdateRT, o_1, o)\ \ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow\ \ \mathbf{node}(o, p, e', n', l)\ conf}$$

(dest-rec-rreq)

$$\frac{\begin{array}{c} RequestID(o, conf) < reqid \\ n' := n[reqID \Rightarrow reqid] \\ m := unicast(rrep(s, o, reqID, pow), o, o_1, conf, (Power(o, conf))^{min}) \end{array}}{\mathbf{msg}(rreq(s, o, reqid, pow), o_1, o)\ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow \mathbf{node}(o, p, e, n', l)\ m\ conf}$$

(rec-rreq1)

$$\frac{\begin{array}{c} o \neq d \\ RequestID(o, conf) < reqid \\ n' := n[reqID \Rightarrow reqid,\ prev \Rightarrow o_1] \\ e' := (power, (energy - power^{min})) \\ m := broadcast(rreq(s, d, reqID, (power + pow)), o, conf, power^{min}) \end{array}}{\mathbf{msg}(rreq(s, d, reqid, pow), o_1, o)\ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow \mathbf{node}(o, p, e', n', l)\ m\ conf}$$

(rec-rreq2)

$$\frac{\begin{array}{c} o \neq d \\ RequestID(o, conf) \geq reqID \end{array}}{\mathbf{msg}(rreq(s, d, reqID, pow), o_1, o)\ conf \longrightarrow conf}$$

Figure 3: The formal model of the routing protocol (1). The variable m denotes new messages added to the configuration. In the rules we assume that $e = (power, energy)$ and skip the trivial premises $power := Power(o, conf)$ and $energy := Energy(o, conf)$.

6.3. A Formal Model of the Power-Sensitive AODV Routing Protocol

The routing protocol discussed in Section 4 is now formalized. The main difference between our protocol and AODV is that we find the *cheapest path* instead of the shortest one. In the model, each node has its own routing table that stores the path to each destination. For each destination, the routing table stores the following information: the next node on the path to the destination and the required power to send data to the destination. When the node finds a cheaper path to a destination (a path which requires less power), it updates its routing table and replaces the old path with the cheaper one. The neighbors of a node are stored in a list *neighbor*.

The rules in Figures 3 and 4 control the message propagation in the model by receiving a route request or a route reply message and sending a

$$\textsc{(rec-rrep1)}$$

$$\frac{\begin{array}{c}(o \neq s)\\ ([d\ o_d\ power_d]) \in Routing(o, conf)\\ findPower(Routing(o, conf), d) \leq pow\\ e' := (power, energy - power^{min})\\ n' := n[reqID \Rightarrow reqid]\\ m := unicast(rrep(s, d, reqid, (power + pow)), o, Prev(o, conf), conf, power^{min})\end{array}}{\mathbf{msg}(rrep(s, d, reqid, pow), o_1, o)\ \ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow \mathbf{node}(o, p, e', n', l)\ m\ conf}$$

$$\textsc{(rec-rrep2)}$$

$$\frac{\begin{array}{c}(o \neq s)\\ ([d\ o_d\ power_d]) \in Routing(o, conf)\\ findPower(Routing(o, conf), d) > pow\\ e' := (power, energy - power^{min})\\ n' := n[reqID \Rightarrow reqid, routing \Rightarrow (updateRouting([d\ o_1\ pow]))]\\ m := unicast(rrep(s, d, reqid, (power + pow)), o, Prev(o, conf), conf, power^{min})\end{array}}{\mathbf{msg}(rrep(s, d, reqid, pow), o_1, o)\ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow \mathbf{node}(o, p, e', n', l)\ m\ conf}$$

$$\textsc{(rec-rrep3)}$$

$$\frac{\begin{array}{c}(o \neq s)\\ ([d\ o_d\ power_d]) \notin Routing(o, conf)\\ e' := (power, energy - power^{min})\\ n' := n[reqID \Rightarrow reqid,\ routing \Rightarrow (routing \cup\ [d\ o_1\ pow])]\\ m := unicast(rrep(s, d, reqid, (power + pow)), o, Prev(o, conf), conf, power^{min})\end{array}}{\mathbf{msg}(rrep(s, d, reqid, pow), o_1, o)\ \mathbf{node}(o, p, e, n, l)\ \ conf \longrightarrow \mathbf{node}(o, p, e', n', l)\ m\ conf}$$

$$\textsc{(src-rec-rrep)}$$

$$\frac{\begin{array}{c}(o = s)\\ n' := n[reqID \Rightarrow reqid,\ routing \Rightarrow (Routing(o, conf) \cup\ [d\ o_1\ pow])]\\ m := \mathbf{msg}(Membership(d), o, o)\end{array}}{\mathbf{msg}(rrep(s, d, reqid, pow), o_1, o)\ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow \mathbf{node}(o, p, e, n', l)\ m\ conf}$$

Figure 4: The formal model of the routing protocol (2). We assume $e = (power, energy)$.

new message which is either a reply or a request. The messages *rreq* (route request) and *rrep* (route reply) include the information of the message source s, destination d, request id *reqid* and the power that is used for routing the message *pow*. The notation $n[y \Rightarrow y']$ denotes that the n-tuple of the node remains unchanged except that the attribute y will change to y'. Furthermore, $power^{min}$ and $power^{max}$ denote the minimum and maximum power that can be used by a node according to the node's *power* value. The function *findPower* extracts the value of required power for data transmission from the routing table (defined in Figure 2).

In Figure 3, the rule DEST-REC-RREQ is enabled when node o that has received the route request message (*rreq*) is the final destination. Node o sends the route reply *rrep* message to the source node s through the previous node in the route, which is node *o1*. The rules REC-RREQ1 and REC-RREQ2 are related to the situation that nodes receive a route request message (*rreq*), but the receiver of the *rreq* is not the destination. In the first rule, the request message's Id is greater than previously seen by the receiver so the message is fresh. Then, the node o updates its information and broadcast it again to find the destination of the route. In the second rule the message has been seen before because the ID of the request message is less than the current message ID, so the message will be ignored.

In Figure 4 the three rules REC-RREP1, REC-RREP2, and REC-RREP3 capture the situation that nodes receive a *rrep* message but the receiver of the route reply message (*rrep*) is not the original sender of the request. Note that in these rules the estimated power consumption is accumulated in the request messages by $power + pow$. In rule REC-RREP1, the routing table remains unchanged because the current route to the destination is cheaper (the smaller power value in the table). Rule REC-RREP2 changes the existing row in the routing table because the new path to the destination is cheaper than the current one. The function *updateRouting*$([d\ o_1\ pow])$ modifies the routing table's entry that is related to the destination d. It updates the next node in the routing path and the transmission power with o_1 and pow. In rule REC-RREP3, no previous path to the destination exists. Therefore, a new row is added to the routing table. The rule SRC-REC-RREP is enabled when the original sender of the request message has received the reply message. This rule sends a *Membership* message that enables another rule to make decisions about the node's membership in the group (see Section 6.4).

6.4. A Formal Model of the Regrouping Protocol

The transition rules of the regrouping protocol are given in Figure 5. Each node which moves to a new location, should inform neighboring leader nodes about its movements. This is done by broadcasting a *Hello* message with node's maximum power when the node has changed position. The START GROUPING rule represents the *Hello* broadcasting (step 1 of the protocol in Section 5).

When a neighboring group leader receives this *Hello* message, a new node has entered the group's signal range. The function *Leaders* returns all the leaders in a configuration, and *Network*$(o_l, conf)$ returns all the members of the leader o_l.

Each message transmission reduces the node's total energy *energy* with respect to the amount of energy that is consumed for sending the message. The leader starts the process to decide whether it is beneficial to accept the new node as a group member based on the power usage in the result

(Start Grouping)

$$\frac{\begin{array}{c} e' := (\mathit{power}, \mathit{energy} - \mathit{power}^{max}) \\ m := \mathit{broadcast}(\mathit{Hello}, o, \mathit{conf}, \mathit{power}^{max}) \end{array}}{\mathbf{msg}(\mathit{Moved}, o, o)\ \mathbf{node}(o, p, e, n, l)\ \mathit{conf} \longrightarrow \mathbf{node}(o, p, e', n, l)\ m\ \mathit{conf}}$$

(Routing Path Request)

$$\frac{\begin{array}{c} o \in \mathit{Leaders}(\mathit{conf}) \\ o_1 \notin \mathit{Network}(o, \mathit{conf}) \\ e' := (\mathit{power}, \mathit{energy} - \mathit{power}^{min}) \\ m := \mathit{broadcast}(\mathit{rreq}(o, o_1, \mathit{RequestID}(o, \mathit{conf}) + 1, 0), o, \mathit{conf}, \mathit{power}^{min}) \end{array}}{\mathbf{msg}(\mathit{Hello}, o_1, o)\ \mathbf{node}(o, p, e, n, l)\ \mathit{conf} \longrightarrow \mathbf{node}(o, p, e', n, l)\ m\ \mathit{conf}}$$

(Membership Decision)

$$\frac{\begin{array}{c} \mathit{joinGroup}(f) \\ f := \mathit{findPower}(\mathit{Routing}(o, \mathit{conf}), o_1) \\ u := \mathit{Utility}(o, \mathit{conf}) \\ e' := (\mathit{power}, \mathit{energy} - \mathit{power}^{max}) \\ m := \mathit{unicast}(\mathit{Invitation}(\mathit{newUtility}(u, f), u^n), o, o_1, \mathit{conf}, \mathit{power}^{max}) \end{array}}{\mathbf{msg}(\mathit{Membership}(o_1), o, o)\ \mathbf{node}(o, p, e, n, l)\ \mathit{conf} \longrightarrow \mathbf{node}(o, p, e', n, l)\ m\ \mathit{conf}}$$

(Grouping)

$$\frac{\begin{array}{c} \mathit{bestGroup}(l, u_1, u, \mathit{Position}(o_1, \mathit{conf}), \mathit{power}) \\ u := \mathit{Utility}(o, \mathit{conf}) \\ e' := (\mathit{power}, \mathit{energy} - (\mathit{power}^{max} + \mathit{power}^{min})) \\ m := \mathit{unicast}(\mathit{Accept}(u_1), o, o_1, \mathit{conf}, \mathit{power}^{min})\ \mathit{unicast}(\mathit{Leave}(u), o, l, \mathit{conf}, \mathit{power}^{max}) \end{array}}{\mathbf{msg}(\mathit{Invitation}(u_1), o_1, o)\ \mathbf{node}(o, p, e, n, l)\ \mathit{conf} \longrightarrow\ \mathbf{node}(o, p, e', n, o_1)\ m\ \mathit{conf}}$$

Figure 5: A formal model of the grouping protocol. We assume $e = (\mathit{power}, \mathit{energy})$.

path. The leader first runs the power-sensitive AODV protocol (presented in Section 6.3) with minimum power to find the cheapest path to the new node by performing *RoutingPathRequest* rule (step 2 of the protocol in Section 5).

If a path is found, the modified AODV protocol ends by letting the leader send a *Membership* message to itself. This message starts the decision making process about the node's membership, which is captured by the *MembershipDecision* rules (step 3 of the protocol in Section 5). The function *newUtility* calculates the new value of the utility function after joining the node, and the function *joinGroup* represents the computation of the utility function (Formula 2), formalized as in *joinGroup* rule.

In *joinGroup* function, e is the total power consumed in the routing path, and power^{max}, i, *rate*, and *pack*, are constants reflecting the maximum sending power, the transmission rate, and the packet size, respectively. These constants can be seen as network parameters, and suitable values given as parameters to the initial configuration. The output of *joinGroup* is a Boolean value. The leader uses this function to decide if a new node could be added as a member.

function $newUtility : Nat \times Nat \longrightarrow Nat$
$newUtility(u, e) = u + ((rate/e) * ((1 - 2.71^{(0.5*i)})^{pack}))$

function $joinGroup : Nat \longrightarrow Bool$
$joinGroup(e) = ((rate/e) * ((1 - 2.71^{(0.5*i)})^{pack})) > ((rate/power^{max}) * ((1 - 2.71^{(0.5*i)})^{pack}))$

function $bestGroup : List[List[Nat]] \times (Nat \times Nat) \times (Nat \times Nat) \times Nat \times Nat \to Bool$
$bestGroup(l, u_1, u, p, power) =$ **if** $inrange(l, p, power)$ **then** $abs(u_1^n - u_1^o) > abs(u^n - u^o)$ **else** $true$

Figure 6: Auxiliary functions for the grouping protocol.

If the leader decides to add the node, it sends an invitation message to the node. The node may receive several invitation messages in case of multiple groups, therefore it should choose one of membership offers that is best for it and also the network. The *Grouping* rule represents the node's behavior after receiving the invitation. Here, *bestGroup* is a function which compares the different membership offers. In this model, like the real environment, messages are queued and received one by one. So, each time we just need to compare two offers (step 4 of the protocol in Section 5).

The *bestGroup* function is used in the the GROUPING rule to compare the difference between the utility that is gain by joining to the new and previous group and accept the new offer if it increase the utility of the new group more than the previous group. Inputs to this function are the leader l, new utility value, current utility value, and also location and power of the node. If the node decides to change to the new group, it will inform the new and previous groups' leaders by sending *Accept* and *Leave* messages (steps 5 and 6 of the protocol in Section 5).

An SOS formalization of a distributed agent system would normally contain a context and a concurrency rule. In our SOS formalization, we cannot add these rules directly, due to the broadcasting function which needs to consider the whole configuration in order to find all nodes within reach. However, the broadcasting mechanism is simulating the network itself, rather than the grouping algorithm. The other parts of the algorithm would allow a context and a concurrency rule. The implementation in rewriting logic discussed below will solve this problem.

7. Implementation and Analysis in Maude

In order to implement and analyze the proposed protocol, we transform the SOS style model to rewriting logic (RL) [6], resolving the declarative aspects of the SOS model. The rewriting logic specification can be seen as a high-level implementation of the protocol; it can be executed on the rewriting logic tool Maude, which provides a range of model exploration facilities [7]. In the rewriting logic, rewrite rules apply to equivalence classes of terms. Thus, when auxiliary functions are needed in the semantics, these are defined in equational logic, and are evaluated in between the state transitions [6]. Consequently, the function definitions of our SOS model are represented by equations in the rewriting logic model, and the transition rules are transformed into rewrite rules. If rewrite rules apply to non-overlapping subconfigurations, the transitions may be performed in parallel. Consequently, concurrency is implicit in RL. Conditional rewrite rules $t \longrightarrow t'$ **`if`** `cond` are allowed, where the condition `cond` is a conjunction of rewrites and equations that must hold for the main rule to apply. In the rules of our grouping protocol, the left hand sides will be subconfigurations, meaning that the rules apply to a part of a system configuration (and therefore describe concurrent behavior) as opposed to the SOS semantics, and the context and concurrency rules are built-in since non-overlapping rewrites may happen simultaneously in RL. In order to handle broadcasting, the rules doing broadcasting are formulated as equations $\{conf\} = \{conf'\}$ where $\{\_\}$ is an operator making a system configuration from a configuration. Thus only these equations are locking the whole configuration, whereas all rules may be applied in parallel. Thus the Maude version reflects the intended concurrent execution of nodes.

The Random Moving rule in Section 6 cannot be directly reformulated in RL, because the new position p' does not occur in the left-hand-side. Therefore we add explicit messages ($DMove(p)$) to tell a node that it should move to position p. These messages are typically part of the initial configuration. This approach allows initial configurations to express different mobility scenarios controlling the movement of nodes, which is needed when testing the protocol with Maude's model checking facilities. With the rule *DirectMoving* below a node can move directly to a desired location, i.e., changing the location of the node in one step.

$$\text{(Direct Moving)}$$
$$\mathbf{msg}(DMove(p'), o, o)\ \mathbf{node}(o, p, e, n, l)\ conf \longrightarrow \mathbf{node}(o, p', e, n, l)\ \mathbf{msg}(Moved, o, o)\ conf$$

In a more realistic model, a node can move to a desired location through a non-deterministic path. This is implemented by the rule *Moving* which makes non-deterministic but finite steps towards the desired destination, issuing a *Moved* message when reaching it, thereby triggering the regrouping protocol.

$$\textsc{(Moving)}\quad \frac{p_o \neq p \qquad p' \in p_o + \{(-1,0),(0,-1),(0,1),(1,0)\} \qquad \mathbf{distance}(p',p) < \mathbf{distance}(p_o,p)}{\mathbf{node}(o,p_o,e,n,l)\ \mathbf{msg}(Move(p),o,o)\ conf \longrightarrow \mathbf{node}(o,p',e,n,l)\ \mathbf{msg}(Move(p),o,o)\ conf}$$

$$\textsc{(Moving Done)}\quad \mathbf{node}(o,p,e,n,l)\ \mathbf{msg}(Move(p),o,o)\ conf \longrightarrow \mathbf{node}(o,p,e,n,l)\ \mathbf{msg}(Moved,o,o)\ conf$$

For correspondence with the previous part of the paper, the rules are still presented in SOS style. All the rules in Section 6 are transformed into rewrite rules, resulting in a Maude implementation of the protocol. More technical details of the Maude model could be found in [3] and [4].

Maude provides model checking tools to check desired properties of a model and a search tool that searches through all reachable states while checking given properties. Maude provides different tools for testing and validating the model. It can also run the model through one path of the state space like a simulator. For simplicity in the implementation (as well as in the SOS model), we have assumed that there is no message loss in the protocol (apart from messages to nodes out of range), that messages do not expire, and that the topology of the network consists of a fixed number of nodes, but nodes can move.

7.1. A case study

As a case study, we consider a topology with six nodes. In this topology, the nodes `b` and `f` are leaders of different groups. We simulate our model with different initial states. Each initial state consists of a topology of the network and a set of node movements. In our analysis the topology was the same for all the initial states, but the movements of the nodes were different. The set of movements includes the following cases:

- Movement of one node in each run that is repeated for different nodes regardless of being a normal node or leader, in the area of the same group or to the range of another group, and also out of the range of any group.
- Simultaneous movement of two nodes in each run. Several permutation of node movements were considered in this part of the validation, including moving nodes to the same group or different groups.

```
⟨ "a":Node | id:1,leader:[2 1 1],neighbors:noneOids,xPos:2,yPos:2,
power:1,utility:4,routingTable:[0 0 0],reqid:0,oldUtility:0,energy:870 ⟩
⟨ "b":Node | id:2,leader:[2 1 1],neighbors:("a";"e"),xPos:1,yPos:1,
power: 1,utility:0,routingTable:[0 0 0],reqid:0,oldUtility:0,energy:805 ⟩
⟨ "c":Node | id:3,leader:[6 10 3],neighbors:noneOids,xPos:8,yPos:4,
power:1,utility:5,routingTable:[0 0 0],reqid:1,oldUtility:0,energy:835 ⟩
⟨ "d":Node | id:4,leader:[6 10 3],neighbors:noneOids,xPos:9,yPos:3,
power:1,utility:7,routingTable:[0 0 0],reqid:1,oldUtility:5,energy:820 ⟩
⟨ "e":Node | false,id:5,leader:[2 1 1],neighbors:noneOids,xPos:8,yPos:4,
power:1,utility:0,routingTable:[0 0 0],reqid:1,oldUtility:0,energy:800 ⟩
⟨ "f":Node | id:6,leader:[6 10 3],neighbors:("c";"d"),xPos:10,yPos:3,
power:1,utility:7, routingTable:([4 4 1][0 0 0]),reqid:1,oldUtility:0,energy:765 ⟩
```

Figure 7: The final state of a Maude simulation of the protocol.

We simulated the model and analyzed the final states to find out if the model behaves correctly. The correctness is formalized as follows:

$$\Box\neg(\mathit{Member}(O, L) \Leftrightarrow \mathit{UEnh}(L) \geq 0) \tag{3}$$

$$\Box\neg(\mathit{Member}(O, L) \Leftrightarrow (\mathit{UEnh}(L) \geq \mathit{UEnh}(i)\ \forall i \in \mathit{Leaders}(\mathit{conf}))) \tag{4}$$

The predicate $\mathit{Member}(O, L)$ is true if node O is a member of the group of leader L. The predicate $\mathit{UEnh}(L)$ gives the utility enhancement if the node joins, i.e., the difference of the new utility of the group of L and its previous utility, considering all nodes in the group which the node chooses to join. Formula 3 means that in all states of the system, the membership of the node in the group is accepted by the leader only when this membership is beneficial for the group and enhances the utility. In other words, there is no state in which the utility value decreases but the node's membership is accepted. Formula 4 demonstrates that if node o participates in the group of leader l, this membership enhances the utility of this group more than the other groups.

To exemplify, consider the scenario in which nodes `c` and `d` change their location such that node `c` stays and `d` comes within the range of the leader `f`. We first use Maude to check this property by simulating the model. The result of the simulation is given in Figure 7. In the Maude implementation, the attributes of a node are presented in a flat manner compared to the SOS model. However, the names should suggest the meaning. The `neighbors` attribute shows the neighbor nodes that are chosen to join the group. By inspecting the `neighbors` attribute of leader `f`, we see that node `c` and `d` are now neighbors of `f`'s group. For these simulations, the results showed that the model works as expected in all the cases.

Although we did several simulations to improve the trustworthiness of the results, simulation can not prove the correctness of the model because it just checks one path in the system's state space, whereas to prove the validity of the model all the possible paths of the state space should be checked for failure. To achieve this goal, we search through all possible states of our model using Maude's search command for a number of given initial states. The search results show that for all possible traces from the initial states the model works correctly. For example, when node `d` (id=4) moves to the position(10,3) which is closer to leader `f`, search proves that in all possible final states, node `d` is a neighbor of leader `f` and there is no case of failure.

In addition, we have analyzed the effects of the grouping protocol on the energy consumption of the WSNs. For this purpose, Maude's simulation tool is used repeatedly. In the beginning of the model execution, the nodes start sending data messages. During the execution, they can move and join a new group. We ran simulations for two distinct scenarios, namely, when the WSN uses the grouping protocol vs. when it does not. Our purpose is to compare the power consumption of the nodes and the leaders, in each separate scenario. The network's architecture could be designed to provide low cost communications between leaders and sink nodes, such as in MULE-based architecture for WSNs [17]. Therefore, we assume that the leaders use the minimum transmission power to send messages to sink nodes.

Figure 8 and Figure 9 represent the saved energy of a sensor node and of a leader, with (red) and without (blue) using the grouping protocol. To generate each of the graphs in the figures, we ran 5 simulations, each simulation lasting for 1000 time units (letting one time unit correspond to one rewrite step). In the initial configuration, an initial value is assigned to the total energy of each node. After each message transmission, including data messages that are sent regularly by the nodes, as well as messages that are related to the grouping protocol, the value of total energy is modified and captured by the graphs. The final result is the average of the results of all the simulations. These results show that for a normal sensor node, it is always beneficial to join a group. Even for a leader, using its minimum transmission power for all of its communications, it is more beneficial to be in a group than to be a separate node, sending only its own messages but using the maximum transmission power.

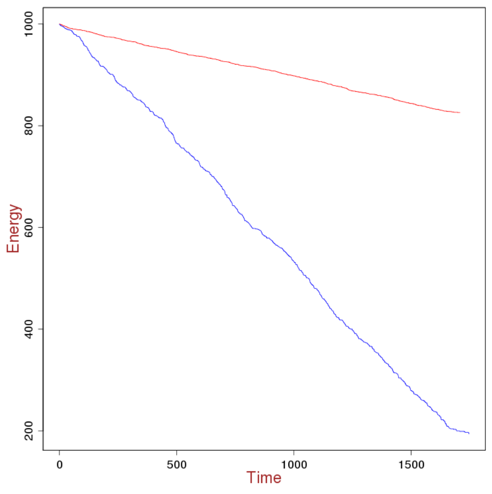

Figure 8: The remaining energy of a node.

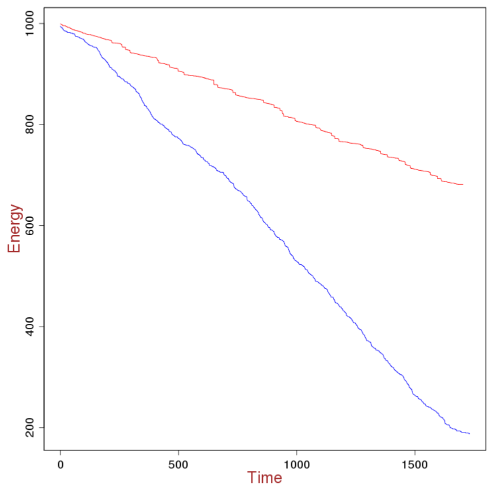

Figure 9: The remaining energy of a leader.

8. Conclusion

In this paper, we propose a group membership protocol for WSNs to choose the best available group by each node. In this protocol, members cooperate with each other to transmit data, in order to decrease the total power consumption of the group and also of the network. A node may move toward the range of several groups that have overlapping range. It should choose the best group to join, applying coalitional game theory with respect to the total power consumption. We present an abstract formal model of the protocol in the SOS framework. The implementation was done by transforming the SOS model of the protocol into rewriting logic, and Maude was used to analyze its behavior for several scenarios.

In future work, we intend to build on our current Maude model as well as extending the model to capture real-time aspects of WSNs. Furthermore, we plan to refine the utility function used in this paper, i.e., to capture the interference of the transmission signals of the nodes. We are also going to study how to change the nodes' role in a group and selecting a new leader. In addition, we want to capture the correlation of transmitted data in order to send them more efficiently by considering this correlation in the cooperation of the nodes. Furthermore we plan to do probabilistic model checking in order to statistically show the correctness of the protocol for larger models.

Acknowledgments. The authors gratefully acknowledge valuable comments from the anonymous reviewers.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (4) (2002) 393–422.

[2] J. Lloret, C. E. Palau, F. Boronat, J. Tomás, Improving networks using group-based topologies, Computer Communications 31 (14) (2008) 3438–3450.

[3] F. Kazemeyni, E. B. Johnsen, O. Owe, I. Balasinham, Grouping nodes in wsns using coalitional game theory, in: J. Hatcliff, E. Zucca (Eds.), Formal Techniques for Distributed Systems, Proc. FMOODS/FORTE'09), Vol. 6117 of Lecture Notes in Computer Science, Springer, 2010, pp. 95–109.

[4] F. Kazemeyni, E. B. Johnsen, O. Owe, I. Balasingham, Group selection by nodes in wireless sensor networks using coalitional game theory, in: Proc. 16th Intl. Conf. on Engineering of Complex Computer Systems (ICECCS 2011), IEEE Computer Society Press, 2011, pp. 253–262.

[5] G. D. Plotkin, A structural approach to operational semantics, Journal of Logic and Algebraic Programming 60-61 (2004) 17–139.

[6] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, Theoretical Computer Science 96 (1992) 73–155.

[7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. L. Talcott (Eds.), All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic, Vol. 4350 of Lecture Notes in Computer Science, Springer, 2007.

[8] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, Energy conservation in wireless sensor networks: A survey, Ad Hoc Netw. 7 (2009) 537–568.

[9] Y. Xu, J. Heidemann, D. Estrin, Geography-informed energy conservation for ad hoc routing, in: Proceedings of the 7th annual international conference on Mobile computing and networking, MobiCom '01, ACM, New York, NY, USA, 2001, pp. 70–84.

[10] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, D. Estrin, Networking issues in wireless sensor networks, J. Parallel Distrib. Comput. 64 (7) (2004) 799–814.

[11] Z. Kong, E. M. Yeh, Distributed energy management algorithm for large-scale wireless sensor networks, in: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '07, ACM, New York, NY, USA, 2007, pp. 209–218.

[12] A. Warrier, S. Park, J. Min, I. Rhee, How much energy saving does topology control offer for wireless sensor networks? - a practical study, Comput. Commun. 30 (2007) 2867–2879.

[13] C. Tang, C. Raghavendra, Compression techniques for wireless sensor networks, in: C. S. Raghavendra, K. M. Sivalingam, T. Znati (Eds.), Wireless Sensor Networks, Springer, 2004, pp. 207–231.

[14] Z. Xiong, A. Liveris, S. Cheng, Distributed source coding for sensor networks, Signal Processing Magazine, IEEE 21 (5) (2004) 80 – 94.

[15] D. Chu, A. Deshpande, J. M. Hellerstein, W. Hong, Approximate data collection in sensor networks using probabilistic models, in: Proceedings of the 22nd International Conference on Data Engineering, ICDE '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 48–.

[16] G. Anastasi, M. Conti, A. Passarella, L. Pelusi, Mobile-relay forwarding in opportunistic networks, in: M. Ibnkahla (Ed.), Adaptive Techniques in Wireless Networks, Taylor and Francis, 2008, Ch. 13.

[17] R. C. Shah, S. Roy, S. Jain, W. Brunette, Data mules: Modeling a three-tier architecture for sparse sensor networks, in: IEEE SNPA Workshop, 2003, pp. 30–41.

[18] A. Chakrabarti, A. Sabharwal, B. Aazhang, Using predictable observer mobility for power efficient design of sensor networks, in: Proceedings of the 2nd international conference on Information processing in sensor

networks, IPSN'03, Springer-Verlag, Berlin, Heidelberg, 2003, pp. 129–145.

[19] Z. M. Wang, S. Basagni, E. Melachrinoudis, C. Petrioli, Exploiting sink mobility for maximizing sensor networks lifetime, in: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Volume 09, IEEE Computer Society, 2005.

[20] M. Cardei, D. zhu Du, Improving wireless sensor network lifetime through power aware organization, ACM Wireless Networks 11 (2005) 333–340.

[21] G. Chen, C. Li, M. Ye, J. Wu, An unequal cluster-based routing protocol in wireless sensor networks, Wireless Networks 15 (2009) 193–207.

[22] D. A. Miller, S. Tilak, T. Fountain, "token" equilibria in sensor networks with multiple sponsors, in: Proc. 1st International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), IEEE Computer Society, 2005.

[23] R. Strauss, A. Abedi, Game theoretic power allocation in sparsely distributed clusters of wireless sensors (gpas), in: IWCMC, 2009, pp. 1454–1458.

[24] H. Inaltekin, S. B. Wicker, The analysis of nash equilibria of the one-shot random-access game for wireless networks and the behavior of selfish nodes, IEEE/ACM Trans. Netw. 16 (5) (2008) 1094–1107.

[25] W. Saad, Z. Han, M. Debbah, A. Hjørungnes, T. Başar, Coalitional game theory for communication networks: A tutorial, IEEE Signal Processing Magazine 26 (5) (2009) 77–97, special issue on Game Theory.

[26] H. N. Pham, D. Pediaditakis, A. Boulis, From simulation to real deployments in WSN and back, in: International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'07), IEEE Computer Society, 2007, pp. 1–6.

[27] S. Park, A. Savvides, M. B. Srivastava, SensorSim: a simulation framework for sensor networks, in: A. Boukerche, M. Meo, C. Tropper (Eds.), Proc. 3rd Intl. Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2000), ACM, 2000, pp. 104–111.

[28] S. C. Ergen, M. Ergen, T. J. Koo, Lifetime analysis of a sensor network with hybrid automata modelling, in: C. S. Raghavendra, K. M. Sivalingam (Eds.), Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), ACM, 2002, pp. 98–104.

[29] A. Fehnker, L. van Hoesel, A. Mader, Modelling and verification of the LMAC protocol for wireless sensor networks, in: J. Davies, J. Gibbons (Eds.), Proc. 6th Intl. Conf. on Integrated Formal Methods (IFM'07), Vol. 4591 of Lecture Notes in Computer Science, Springer, 2007, pp. 253–272.

[30] S. Tschirner, L. Xuedong, W. Yi, Model-based validation of QoS properties of biomedical sensor networks, in: L. de Alfaro, J. Palsberg (Eds.), Proceedings of the 8th ACM & IEEE International conference on Embedded software (EMSOFT'08), ACM, 2008, pp. 69–78.

[31] A. Fehnker, M. Fruth, A. McIver, Graphical modelling for simulation and formal analysis of wireless network protocols, in: M. Butler, C. B. Jones, A. Romanovsky, E. Troubitsyna (Eds.), Methods, Models and Tools for Fault Tolerance, Vol. 5454 of Lecture Notes in Computer Science, Springer, 2009, pp. 1–24.

[32] J. S. Dong, J. Sun, J. Sun, K. Taguchi, X. Zhang, Specifying and verifying sensor networks: An experiment of formal methods, in: S. Liu, T. S. E. Maibaum, K. Araki (Eds.), 10th International Conference on Formal Engineering Methods (ICFEM'08), Vol. 5256 of Lecture Notes in Computer Science, Springer, 2008, pp. 318–337.

[33] E. B. Johnsen, O. Owe, J. Bjørk, M. Kyas, An object-oriented component model for heterogeneous nets, in: F. S. de Boer, M. M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Proc. 6th International Symposium on Formal Methods for Components and Objects (FMCO 2007), Vol. 5382 of Lecture Notes in Computer Science, Springer, 2008, pp. 257–279.

[34] S. Nair, R. Cardell-Oliver, Formal specification and analysis of performance variation in sensor network diffusion protocols, in: S. Balsamo, C.-F. Chiasserini, L. Donatiello (Eds.), Proceedings of the 7th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04), ACM, 2004, pp. 170–173.

[35] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, J. Quesada, Towards Maude 2.0, in: 3rd International Workshop on Rewriting Logic and its Applications (WRLA'00), Vol. 36 of Electronic Notes in Theoretical Computer Science, Elsevier, 2000, pp. 294–315.

[36] J. Meseguer, Software specification and verification in rewriting logic (lectures at the 2002 marktoberdorf summer school), in: M. Broy, M. Pizka (Eds.), Models, Algebras and Logics of Engineering Software, Vol. 191 of NATO science series: Computer and systems sciences, IOS Press, 2003, pp. 133–193.

[37] P. C. Ölveczky, S. Thorvaldsen, Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude, Theoretical Computer Science 410 (2-3) (2009) 254–280.

[38] M. Katelman, J. Meseguer, J. C. Hou, Redesign of the LMST wireless sensor protocol through formal modeling and statistical model checking, in: G. Barthe, F. S. de Boer (Eds.), Proc. 10th Intl. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08), Vol. 5051 of Lecture Notes in Computer Science, Springer, 2008, pp. 150–169.

[39] C. E. Perkins, E. M. Belding-Royer, Ad-hoc on-demand distance vector routing, in: WMCSA, 1999, pp. 90–100.

[40] A. Gersho, R. M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[41] D. Fudenberg, J. Tirole, Game Theory, MIT Press, Cambridge, MA, 1991.

[42] T. Başar, G. J. Olsder, Dynamic non-cooperative game theory, SIAM, 1999.

[43] D. Goodman, N. Mandayam, Power control for wireless data, IEEE Personal Communications 7 (2000) 48–54.

[44] A. B. Mackenzie, S. B. Wicker, Game theory and the design of self-configuring, adaptive wireless networks, IEEE Communications Magazine 39 (11) (2001) 126–131.