Grouping Nodes in Wireless Sensor Networks Using Coalitional Game Theory

Fatemeh Kazemeyni[1,2], Einar Broch Johnsen[1],
Olaf Owe[1], and Ilangko Balasingham[2,3]

[1] Department of Informatics, University of Oslo, Norway
{fatemehk,einarj,olaf}@ifi.uio.no
[2] The Interventional Center, Oslo University Hospital, Institute of Hospital Medicine, University of Oslo, Norway
[3] Department of Electronics and Telecommunication, Norwegian University of Science and Technology, Trondheim, Norway
ilangkob@medisin.uio.no

Abstract. Wireless sensor networks are typically ad-hoc networks of resource-constrained nodes; in particular, the nodes are limited in power resources. It can be difficult and costly to replace sensor nodes, for instance when implanted in the human body. Data transmission is the major consumer of power, so it is important to have power-efficient protocols. In order to reduce the total power consumption in the network, we consider nodes which cooperate to transmit data. Nodes which cooperate, form a *group*. A mobile node may at some point be without a group, in which case it is desirable for the node to be able to join a group. In this paper we propose a modification of the AODV protocol to decide whether a node should join a given group, using coalitional game theory to determine what is beneficial in terms of power consumption. The protocol is formalized in rewriting logic, implemented in the Maude tool, and validated by means of Maude's model exploration facilities.

1 Introduction

A wireless sensor network (WSN) often contains hundreds or thousands of sensor nodes equipped with sensing, computing, and communication devices such as short-range communication devices over wireless channels. These nodes may be distributed over a large area; e.g., WSNs can do area monitoring for some phenomenon of interest. In such an application, the main goal of the WSN is to collect data from the environment and send it to a sink node. In many cases, WSNs are ad-hoc networks with mobile nodes which need to self-configure.

Sensor nodes are very small and often difficult to replace. This size limitation introduces challenges for the design and management of WSNs; in particular, restrictions in memory, power, and communication capacity need to be considered in order to improve the longevity of the nodes. Power restriction is the most remarkable of these constraints: The range of data transmission depends on the power used by the node. Reduced power consumption is an important goal in the

design of WSN protocols. Because data transmission is expensive, the management of communication between nodes plays a vital role for the power efficiency of these networks. *Cooperation* between sensor nodes can potentially reduce the total power consumed for data transmission in the whole network.

Grouping is a method to organize node cooperation in a WSN. A group of nodes has a leader which receives data from the group members and communicates with the outside of the group. 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 a 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, a new node may want to join a group and the group needs to decide whether to accept the node.

This paper proposes a protocol to decide whether a node should join a group. We assume that a group leader forms and manages its group in a way that is beneficial for the group's members, and that a node can transmit directly to the group leader using maximum transmission power. Our protocol uses coalitional game theory to decide on group extensions. Adapting the AODV routing protocol for ad-hoc networks [21], a utility function over the dynamically determined route from the new node to the leader decides whether it is beneficial in terms of power consumption that the new node joins the group. We develop a formal, executable model of the proposed protocol in rewriting logic [16]. The resulting model is validated using Maude [3], some initial results are presented here.

Related work. 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 [22] and SensorSim [20].

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 [5] and UPPAAL has been applied to the LMAC protocol [7] and to the temporal configuration parameters of radio communication [25]. The CaVi tool combines simulation in Castalia with probabilistic model checking [6]. A recent process algebra for active sensor processes includes primitives for, e.g., sensing [4]. A Creol extension for heterogeneous environments includes radio communication [12]. The Temporal Logic of Actions has been used for routing tree diffusion protocols [18].

Ölveczky and Thorvaldsen have shown 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 [19]. Their approach has also been combined with probabilistic model-checking to analyze the LMST protocol [13]. We follow

this line of research and use Maude as a tool to develop a grouping protocol [14] for WSNs, applying coalitional game theory to estimate power consumption. Noncooperational game theory has been used to reduce the power consumption of sensor nodes, applying a utility function to find the Nash equilibrium [11, 17, 24]. Coalitional game theory is applied to reduce the power consumption in WSNs by [23], who propose a merge and split approach for coalition formation. They 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.

Paper overview. Section 2 introduces WSNs and grouping and Section 3 presents coalitional game theory. Section 4 proposes a group membership protocol based on coalitional game theory. Section 5 briefly summarizes rewriting logic and Maude, used to develop a formal model of the protocol in Section 6 and for analysis of an example topology in Section 7. 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.

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.

Wireless sensor nodes use routing protocols to communicate with each other and to find the path to the designated sink node (or nodes) in order to transmit the data that is sensed from the environment. In most of these protocols, the nodes broadcast their data to all nodes that are within their data transmission range. 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 [21], 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.

Grouping is a method for cooperation between nodes, in which nodes belong to distinct groups [14]. 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 and this role can be exchanged between group members at specific time intervals. In this paper we do not focus on the leader selection issue but rather on the group management.

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. The group formation could be done by using different techniques. In general, the grouping of nodes can be done based on special characteristics or distance. In the first case, a special correlation among the sensors should be found by using vector quantization [9]. For example, all the sensor nodes that have similar sensed data could be placed in one group. In the second case, the sensor nodes are formed in 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.

The AODV Routing protocol. The Ad-hoc On-Demand Distance Vector (AODV) routing protocol [21] is a reactive protocol; 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 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 routing table entries. The 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 hop count which shows the length of 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.

3 Coalitional Game Theory

Game theory [8] can be used to analyze behavior in decentralized and self-organizing networks. Game theory models the actions and 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 divided into noncooperative [2] and cooperative game theory [8]. Noncooperative game theory studies the interaction between competing players, where each player chooses its strategy independently and each player's goal is to improve its utility or reduce its cost [23]. In contrast, cooperative (or coalitional) game theory considers the benefit of all the players. In coalitional games, players choose the strategies to maximize the utility for all players. In these games, cooperating groups of players are formed, called coalitions. Coalitional games are useful to design fair, robust, and efficient cooperation strategies in communication networks [23].

In a coalitional game (N, v) with N players, the utility of a coalition is determined by a *characteristic function* $v : 2^N \rightarrow \mathbb{R}$ which applies to coalitions of players. For a coalition $S \subseteq N$, $v(S)$ depends on the members of S, but not on the structure of the 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 many 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 [23].

4 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.* [10] appears to be a suitable choice when power consumption is an important factor of the model [15]:

$$u(P_j, \delta_j) = (\frac{R}{P_j})(1 - e^{-0.5\delta_j})^L. \quad (1)$$

When applying u to a node j, P_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. For simplicity, we assume that the SINR is fixed and the same for all the nodes.

Wireless sensor nodes can transfer data with different amounts of power. Let P^{max} denote the maximum transmission power and P_j^{min} the minimum power for each node j, such that $0 \leq P_j^{min} \leq P^{max}$. When a node j cooperates in a group, it uses P_j^{min} for message transmission, and otherwise P^{max}. Consider a network of nodes $N = \{1, \ldots, n\}$. When there is no coalition between nodes in N, we have

$$\sum_{j=1}^{n} u(P_j, \delta) = \sum_{j=1}^{n} u_j(P^{max}, \delta).$$

In contrast, if all the nodes in N cooperate, we have:

$$\sum_{j=1}^{n} u(P_j, \delta) = \sum_{j=1}^{n} u(P_j^{min}, \delta).$$

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:

$$\sum_{j=1}^{n} u(P_j^{min}, \delta) > \sum_{j=1}^{n} u(P^{max}, \delta).$$

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 original sender 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:

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

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 powers consumed by all the nodes in the routing path through which data is transmitted from the sender to the leader.

Using this utility function, the leader can decide about the membership of a new node more precisely and with more realistic estimations. The result of this utility function depends on the specific topology, so coalition is not always beneficial. Depending on the value calculated using Formula 2, node j will become

a member of the group or not. According to the utility function it is more beneficial for the node to follow a path through the group than to act individually, if the following formula holds:

$$u(P^{max}, \delta) < u(P_j, \delta_j) = (\frac{R}{\sum_{n \in RP_{j,Leader}} P_n^{min}})(1 - e^{-0.5\delta_j})^L.$$

We adapt the AODV protocol to find the cheapest routing path between the new node and the leader in terms of power usage. In AODV, each node uses its maximum power to send all its messages, including `hello`, `rreq`, and `rrep` messages. In order to faithfully capture the real environment, the leader and the node should be in the signal range of each other. Consequently, the node can communicate directly with the leader by using its maximum transmission power. However, inside the group it is sufficient to use the minimum power for data transmission. Consequently, we let nodes transmit messages with minimum power when running the AODV routing protocol. In this situation, AODV can only find the routing path between the new node and the group leader in the case where coalition is possible. However, the shortest path in terms of hops is not necessarily the cheapest path when the minimum transmission power of nodes may vary. Therefore, we modify the AODV protocol to accumulate the needed power consumption along the path instead of the number of hops. The details are given in Section 6.

5 Rewriting Logic and Maude

The formal model of the protocol is defined in rewriting logic (RL) [16] and can be analyzed using Maude [3]. A rewrite theory is a 4-tuple (Σ, E, L, R) where the signature Σ defines the function symbols, E defines equations between terms, L is a set of labels, and R is a set of labeled rewrite rules. Rewrite rules apply to terms of given sorts. Sorts are specified in (membership) equational logic (Σ, E). When modeling computational systems, different system components are typically modeled by terms of suitable sorts defined in the equational logic. The global state configuration is defined as a multiset of these terms. RL extends algebraic specification techniques with transition rules: The dynamic behavior of a system is captured by rewrite rules supplementing the equations which define the term language. From a computational viewpoint, a rewrite rule $t \longrightarrow t'$ may be interpreted as a *local transition rule* allowing an instance of the pattern t to evolve into the corresponding instance of the pattern t'. When auxiliary functions are needed in the semantics, these are defined in equational logic, and are evaluated in between the state transitions [16]. If rewrite rules apply to non-overlapping sub-configurations, 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 Maude, equations are denoted by **`eq`**, labeled rules by **`rl`** `[name]`, conditional ones by **`ceq`** and **`crl`**, respectively, and variable names are capitalized. Given an initial state of a model,

Maude supports simulation as well as breadth-first search through reachable states and model checking of finite reachable states for desired properties.

6 Modeling and Validation

In this section, we define a formal model of the group membership protocol in rewriting logic. In the model, we assume that there is no message loss in the protocol and that the topology of the network consists of a fixed number of nodes, but nodes can move and messages do not expire.

A *system configuration* is a multiset of objects and messages inside curly brackets. Following RL conventions, whitespace denotes the associative and commutative constructor for configurations. The term ⟨ O : Node | Attributes ⟩ denotes a Node object, where O is the identifier and Attributes is a set of attributes of the form Attr : X. Here, Attr is an attribute name and X the associated value. Node objects have the following attributes: the Boolean leader? indicates if the node is a leader, leader is a list that stores the information of the node's leader, the set neighbors contains the neighbor nodes, and the set members contains the group members of a leader node. The natural numbers xLoc and yLoc contain the horizontal and vertical position of the node, respectively. The natural number power stores the current sending power level of the node. The routes is a list of routes; i.e., of lists consisting of a destination ID, the next node in the path to the destination and the accumulated power that is required to send data to the destination. The natural number reqid stores the identifier of the last received message and pred the identity of the last neighbor that communicated with the node (the predecessor in the protocol).

We now explain the rules and equations modeling wireless message passing, node movement, the routing protocol, and the evaluation of the utility function.

6.1 Unicast and Broadcast

Unicast messages have the form (M **from** O X Y P **to** O') where M is the messages body (possibly with parameters), O the source with current location (X,Y), O' the destination, and P the sending power used. A message will not reach its destination unless it is within the range. This is modeled by the equation

```
ceq (M from O X Y P to O´) ⟨ O´: Node | xLoc: X´, xLoc: Y´, A ⟩
  = ⟨ O´: Node | xLoc: X´, xLoc: Y´, A ⟩ if not inrange(X,Y,X´,Y´,P).
```

where inrange is a Boolean function checking that the two locations (X,Y) and (X',Y') are in range of each other with power P (using the calculated distance and the network parameters including the interference level). Note that this equation removes a message which cannot reach its destination, depending on the location values at sending time.

Transmission is modeled by a rule which creates a message, the equation above (if enabled) and a rule consuming the message if it is in range. This reflects the time when a message is queued for sending, the transmission time (immediate for wireless communication) and the queued at the destination, and

the time the message is taken out of the destination queue. Thus it models the immediate transmission mode of wireless communication, and it allows several nodes to move around at the same time.

Multicasting is modeled by allowing a multiset of destinations and the following equations which expand the destination list:

```
eq (M from O X Y P to noneOids) = none .
eq (M from O X Y P to O'; Os) =
     (M from O X Y P to O') (M from O X Y P to Os).
```

Here, `Os` denotes a multiset of object identities (with ";" as multiset constructor).

Wireless broadcasting uses messages `(M from O X Y P to all)` where `all` is a constructor indicating that it is sent to all nodes within range. The use of `all` is defined by the equation

```
eq {C (M from O X Y P to all)} = {C (M from O X Y P to nodes(C)-O)} .
```

Here, `nodes(C)` gives the multiset of all node identities in the configuration `C`. Thus broadcasting is reduced to multicasting. Due to the first equation above only nodes in the range can receive the message. This equation models the underlying network and therefore applies to the whole system.

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 nodes. We have modeled three different methods for node movement. In the first method, the node can move freely everywhere, captured by rules that non-deterministically change the location of the node. In the second one, a node can move directly to a desired location. A rule will change the location of the node in one step. In the last method, a node can move to a desired location through a non-deterministic path, there are rules that determine the final destination and do non-deterministic but finite steps toward it. In each step, the vertical or horizontal location of the node is decreased or increased by one unit, but rules always check that the new location is closer to the desired location before the node moves.

6.3 The Regrouping

Each node should inform neighboring leader nodes about its movements. This is done by broadcasting a `hello` message with maximum power when it has changed position. The following rule represents the `hello` broadcasting:

```
rl [moving-done] :(movemsg Xn Yn from O X Y P to O)
⟨O :Node | xLoc: Xn, yLoc: Yn, neighbors: N, power: P, A⟩
⟶ ⟨O :Node | xLoc: Xn, yLoc: Yn, neighbors: N, power: P, A⟩
(hello from O X Y Pmax to all) .
```

When a neighboring group leader receives this `hello` message, a new node has entered the group's signal range. The leader will then start the process to decide whether the new node should be accepted as a group member based on the power information in the result path. The leader first runs the modified

AODV protocol (presented below in Section 6.4) with minimum power to find the cheapest path to the new node. If a path is found, the modified AODV protocol ends by letting the leader send a message `membershipMsg` to itself. This message starts the decision making process about the node's membership, which is captured by the following rules:

```
crl [Membershipdecision] :(membershipMsg Oc from O' X' Y' P' to O)
⟨ O :Node | leader?: true, members: Os, xLoc: X, yLoc: Y, routes: RT,
      power: P, A ⟩ ⟶
⟨ O :Node | leader?: true, members: (Oc ; Os), xLoc: X, yLoc: Y,
      routes: RT, power: P, A ⟩ (join from O X Y P to Oc)
if joinGroup(findPower(RT)).

rl [Join] :(join from O X Y P to Oc) ⟨ Oc: Node | leader: [ O' X' Y' ], A ⟩
⟶ ⟨ Oc: Node | leader: [ O X Y ], A ⟩ .
```

where the function `findPower` extracts the value of required power for data transmission from the routing table, and the function `joinGroup` represents the computation of the utility function (Formula 2), formalized as follows:

```
op joinGroup : Nat → Bool .
eq joinGroup (P) = (RATE quo P * ((1 - 2.71 ^ (0.5 * I)) ^ PACK)
             > (RATE quo Pmax) * ((1 - 2.71 ^ (0.5 * I)) ^ PACK).
```

Here, `P` is the total power consumed in the routing path, and `Pmax`, `I`, `RATE`, and `PACK`, are constants reflecting the maximum sending power, the signal interference level, 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 should be added as a member. The `Join` rule may be adjusted to enable optimal selection in case of multiple group membership offers.

6.4 The 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 `neighbors`. Fig. 2 shows a simplified graph of our model.

All the messages in the routing protocol are modeled as messages in Maude and behave as explained in Section 6.1. The rules in Fig. 1 control the message propagation in the model by receiving a route request or a route reply message and sending a new message which is either a reply or a request. The rule `dest-rec-rreq` is enabled when the node that has received the request is the final destination. The next two rules, both named `rec-rreq`, are related and require that the receiver of the request message is not the destination. In

```
crl [dest-rec-rreq] :(singlerreq SID DID ID P from O X Y Pj to O')
⟨O' :Node|reqid: I, power: POW, xLoc: X', yLoc: Y', A⟩
⟶ ⟨O' :Node|reqid: ID, power: POW, xLoc: X', yLoc: Y', A⟩
(rrep SID DID ID POW from O' X' Y' POW to O) if O' =DID .

crl [rec-rreq1] :(singlerreq SID DID ID P from O X Y Pj to O')
⟨O' :Node|pred: O1, neighbors: N, reqid: I, power: POW, xLoc: X',
      yLoc: Y', A⟩
⟶ ⟨O' :Node|pred: O, neighbors: N, reqid: ID, power: POW, xLoc: X',
      yLoc: Y', A⟩ (rreq SID DID ID (P +POW) from O' X' Y' POW to N)
if (O' ≠DID) ∧ (I < ID) .

crl [rec-rreq2] :(singlerreq SID DID ID P from O X Y Pj to O')
⟨O' :Node|pred: O1, neighbors: N, reqid: I, A⟩
⟶ ⟨O' :Node|pred: O, neighbors: N, reqid: ID, A⟩
if (O' ≠DID) ∧ (I ≥ID) .

crl [rec-rrep1] :(rrep SID DID ID P from O X Y Pj to O')
⟨O' :Node|routes: RT, pred: O1, neighbors: N, power: POW,
      xLoc: X', yLoc: Y', A⟩
⟶ ⟨O' :Node|routes: RT, pred: O1, neighbors: N, power: POW, xLoc: X',
      yLoc: Y', A⟩ (rrep SID DID ID (P +POW) from O' X' Y' POW to O1)
if (O' ≠SID) ∧ (findPower(RT, DID) > P) .

crl [rec-rrep2] :(rrep SID DID ID P from O X Y Pj to O')
⟨O' :Node|routes: (RT [ DID X Y ] RT'), pred: O1, neighbors: N,
      power: POW, xLoc: X', yLoc: Y', A⟩
⟶ ⟨O' :Node|routes: (RT [ DID O P ] RT'), pred: O1,
      neighbors: N, power: POW, xLoc: X', yLoc: Y', A⟩
(rrep SID DID ID (P +POW) from O' X' Y' POW to O1) if (O' ≠SID)
∧ (findPower((RT [ DID X Y ] RT'), DID) < P)∧(findPower(RT, DID)≠ 0).

crl [rec-rrep3] :(rrep SID DID ID P from O X Y Pj to O')
⟨O' :Node|routes: RT, pred: O1, neighbors: N, power: POW,
      xLoc: X', yLoc: Y', A⟩
⟶ (rrep SID DID ID (P +POW) from O' X' Y' POW to O1)
⟨O' :Node|routes: (RT [ DID O P ]), pred: O1,
      neighbors: N, power: POW, xLoc: X', yLoc: Y', A⟩
if (O' ≠SID) ∧ (findPower(RT, DID) < P) ∧ (findPower(RT, DID) =0) .

crl [src-rec-rrep] :(rrep SID DID ID P from O X Y Pj to O')
⟨O' :Node|routes: RT, pred: O1, neighbors: N, power: POW,
      xLoc: X', yLoc: Y', A⟩
 ⟶ (membershipMsg DID from O' X' Y' POW to O')
⟨O' :Node|routes: (RT [ DID O P ]), pred: O1, neighbors: N,
      power: POW, xLoc: X', yLoc: Y', A⟩ if (O' =SID) .
```

Fig. 1. Rules for message propagation.

the first rule, the request message's `ID` is greater than previously seen by the receiver so the message is fresh. 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. The three following rules represent the situation that the receiver of the reply message is not the original sender of the request. Note that in these rules the estimated power consumption is accumulated in

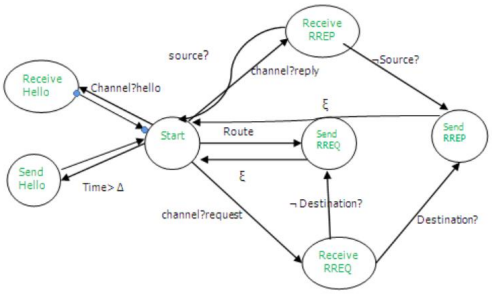

Fig. 2. The graph of the model.

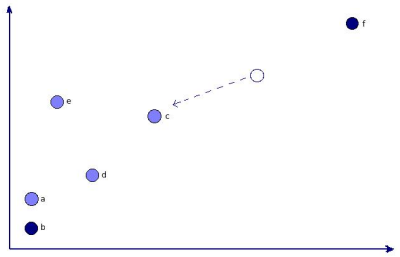

Fig. 3. The topology of the case study.

the request messages by `P + 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 the current one. 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 message `membershipMsg` that enables a rule that make decisions about node's membership in the group (see Section 6.3).

7 Analysis of the Case Study

Maude provides different tools for testing and validation of the model. It can run the model through just one path of the state space like a simulator (using the rewrite command `rew`), which gives us the result of a single run of the model. As a case study, we consider a topology with six nodes such that one of the nodes moves to the range of a group. The topology is shown in Fig. 3. In this topology, the nodes `b` and `f` are leaders of different groups. Nodes `a`, `d`, and `e` are the members of the group with leader `a`, and node `c` is the member of the group with leader `f`. We consider the scenario in which node `c` changes its location such that it comes within the range of the leader `b`. As group leader, node `b` decides about the membership of `c` in the group. The property that we expect from the system is beneficial membership; i.e., the membership of the node in the group is accepted by the leader `b` only when this membership is beneficial for the group. We first use Maude to check this property by simulating the model. The result of the simulation is given in Fig. 4. By inspecting the `members` attributes of leaders `b` and `f`, we see that node `c` now is a member of `b`'s group, while that of `f` is empty.

Simulation can not prove the correctness of the model because it just checks one path in the system's state space, to prove the validity of the model all the possible paths of the state space should be checked for failure. To prove the validity of the model, we search for all possible final states of our model using Maude's

```
{⟨ "a" :Node | leader?: false, leader: [2 1 1], neighbors:
   ("b" ; "c" ; "d"), members: noneOids, xLoc: 1, yLoc: 1,
    routes: ([0 0 0] [3 3 1]), reqid: 1, pred: "d", power: 2 ⟩
⟨ "b" :Node | leader?: true, leader: [2 1 1], neighbors:
   ("a" ; "c" ; "d"), members: ("a" ; "c" ; "d" ; "e"), xLoc: 2,
    yLoc: 2, routes: ([0 0 0] [3 1 2]), reqid: 1, pred: "d", power: 2 ⟩
⟨ "c" :Node | leader?: false, leader: [2 2 2], neighbors:
   ("a" ; "e"), members: noneOids, xLoc: 6, yLoc: 8, routes:
    [0 0 0], reqid: 1, pred: "a", power: 2 ⟩
⟨ "d" :Node | leader?: false, leader: [2 1 1], neighbors:
   ("a" ; "b"), members: noneOids, xLoc: 4, yLoc: 3, routes:
   [0 0 0], reqid: 1, pred: "b", power: 1 ⟩
⟨ "e" :Node | leader?: false, leader: [2 1 1], neighbors:
   "c", members: noneOids, xLoc: 9, yLoc: 11, routes: [0 0 0],
    reqid: 0, pred: "a", power: 2 ⟩
⟨ "f" :Node | leader?: true, leader: [6 20 20], neighbors:
   "c", members: noneOids, xLoc: 9, yLoc: 11, routes: [0 0 0],
    reqid: 0, pred: "a", power: 2 ⟩ }
```

Fig. 4. Final state of the protocol for the case study.

search command. The syntax `search initState =>! C:Configuration` indicates that we search through all final states `C` reachable from the initial configuration `initState`. The result of this search is the same as for the simulation (cf. Fig. 4). Furthermore, the search shows that there is no other solution.

In order to validate the model, we use a correctness function. This is the function `joinGroup` that we used in the model to decide about membership of a node. We now search for a final state of the model such that node `c` is a member of the group of leader `b`, but such that the membership of this node is not beneficial for the group. In this case the function `joinGroup` should decide not to add the node to the group. We define a function `findCheapest` which statically finds the cheapest path from a node `j` to the leader in a given configuration `C`. Thus we compare the statically identified cheapest route with the result of running the protocol. If this search finds a state, it means that the model is not correct because a violation of the desired property has occurred. We do this analysis by a search command using a *such that* clause to specify the desired property.

```
search initState ⟶ ! {C:Configuration ⟨ "b" :Node | routes: RT:ListListNat,
members: ("c" ; ML:OidSet), ATTS:AttributeSet ⟩ } such that
  joinGroup (findCheapest (c,b, C:Configuration ⟨ "b" :Node | routes:
    RT:ListListNat, members: ("c" ; ML:OidSet), ATTS:AttributeSet ⟩ ))
      = false .
```

Maude checks all final states, but finds no solution to this search. This analysis demonstrates how Maude can be used to check the correctness of the protocol, limited to one initial state at the time. For a more in-depth analysis of the correctness of the protocol, we need to generate a set of initial states reflecting representative scenarios. However, how to generate such a set is beyond the scope of this paper and isleft for future work.

8 Conclusion

In this paper, we propose a protocol to decide on group membership for WSNs with mobile nodes. This work is done in the context of a cooperation project with the national hospital of Norway, where wireless technology is developed for medical applications. This paper addresses a subproblem which is common to several grouping protocols. Group members cooperate with each other to transmit data, in order to decrease the total power consumption of the group. When a node moves, it may need to join a new group and coalitional game theory is applied to find the best groups with respect to the total power consumption. The protocol we propose combines a modified version of the AODV protocol with coalitional game theory in order to find the cheapest route in a group with respect to power consumption. We have formalized the protocol in rewriting logic and used Maude to analyze its behavior for an example scenario.

In future work, we intend to refine the utility function used in this paper, in particular to capture the interference of the transmission signals. Other interesting extensions of our current work are criteria for the dynamic merging of groups and topologies in which the leader may change. For these problems, we intend to build on our current Maude model and to extend the model to capture real-time aspects of WSNs. Orthogonally to these extensions, we plan to strengthen our analysis by developing a broader base of representative WSN scenarios.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.
2. T. Başar and G. J. Olsder. *Dynamic non-cooperative game theory.* SIAM, 1999.
3. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285:187–243, Aug. 2002.
4. J. S. Dong, J. Sun, J. S. 0001, K. Taguchi, and X. Zhang. Specifying and verifying sensor networks: An experiment of formal methods. In S. Liu, T. S. E. Maibaum, and K. Araki, editors, *10th Intl. Conf. on Formal Engineering Methods (ICFEM'08)*, volume 5256 of *LNCS*, pages 318–337. Springer, 2008.
5. S. C. Ergen, M. Ergen, and T. J. Koo. Lifetime analysis of a sensor network with hybrid automata modelling. In C. S. Raghavendra and K. M. Sivalingam, editors, *Proc. First ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 98–104. ACM, 2002.
6. A. Fehnker, M. Fruth, and A. McIver. Graphical modelling for simulation and formal analysis of wireless network protocols. In M. Butler, C. B. Jones, A. Romanovsky, and E. Troubitsyna, editors, *Methods, Models and Tools for Fault Tolerance*, volume 5454 of *LNCS*, pages 1–24. Springer, 2009.
7. A. Fehnker, L. van Hoesel, and A. Mader. Modelling and verification of the LMAC protocol for wireless sensor networks. In J. Davies and J. Gibbons, editors, *Proc. 6th Intl. Conf. on Integrated Formal Methods (IFM'07)*, volume 4591 of *LNCS*, pages 253–272. Springer, 2007.
8. D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, Cambridge, MA, 1991.

9. A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.
10. D. Goodman and N. Mandayam. Power control for wireless data. *IEEE Personal Communications*, 7:48–54, 2000.
11. H. Inaltekin and 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):1094–1107, 2008.
12. E. B. Johnsen, O. Owe, J. Bjørk, and M. Kyas. An object-oriented component model for heterogeneous nets. In F. S. de Boer, M. M. Bonsangue, S. Graf, and W.-P. de Roever, editors, *Proc. 6th Intl. Symp. on Formal Methods for Components and Objects (FMCO 2007)*, volume 5382 of *LNCS*, pages 257–279. Springer-Verlag, 2008.
13. M. Katelman, J. Meseguer, and J. C. Hou. Redesign of the lmst wireless sensor protocol through formal modeling and statistical model checking. In G. Barthe and F. S. de Boer, editors, *Proc. 10th Intl. Conf. on Formal Methods for Open Object-Based Distributed Systems (FMOODS'08)*, volume 5051 of *LNCS*, pages 150–169. Springer, 2008.
14. J. Lloret, C. E. Palau, F. Boronat, and J. Tomás. Improving networks using group-based topologies. *Computer Communications*, 31(14):3438–3450, 2008.
15. A. B. Mackenzie and S. B. Wicker. Game theory and the design of self-configuring, adaptive wireless networks. *IEEE Communications Magazine*, 39(11):126–131, Nov. 2001.
16. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.
17. D. A. Miller, S. Tilak, and T. Fountain. "token" equilibria in sensor networks with multiple sponsors. In *CollaborateCom*. IEEE, 2005.
18. S. Nair and R. Cardell-Oliver. Formal specification and analysis of performance variation in sensor network diffusion protocols. In S. Balsamo, C.-F. Chiasserini, and L. Donatiello, editors, *Proc. 7th Intl. Symp. on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM'04)*, pages 170–173. ACM, 2004.
19. P. C. Ölveczky and S. Thorvaldsen. Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude. *Theor. Comput. Sci.*, 410(2-3):254–280, 2009.
20. S. Park, A. Savvides, and M. B. Srivastava. SensorSim: a simulation framework for sensor networks. In A. Boukerche, M. Meo, and C. Tropper, editors, *Proc. 3rd Intl. Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2000)*, pages 104–111. ACM, 2000.
21. C. E. Perkins and E. M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *WMCSA*, pages 90–100. IEEE Computer Society, 1999.
22. H. N. Pham, D. Pediaditakis, and A. Boulis. From simulation to real deployments in WSN and back. In *Intl. Symp. on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'07)*, pages 1–6. IEEE, 2007.
23. W. Saad, Z. Han, M. Debbah, A. Hjørungnes, and T. Başar. Coalitional game theory for communication networks: A tutorial. *IEEE Signal Processing Magazine*, 26(5):77–97, Sept. 2009. Special Issue on Game Theory.
24. R. Strauss and A. Abedi. Game theoretic power allocation in sparsely distributed clusters of wireless sensors (gpas). In M. Guizani, P. Mueller, K.-P. Fähnrich, A. V. Vasilakos, Y. Zhang, and J. Zhang, editors, *IWCMC*, pages 1454–1458. ACM, 2009.
25. S. Tschirner, L. Xuedong, and W. Yi. Model-based validation of QoS properties of biomedical sensor networks. In L. de Alfaro and J. Palsberg, editors, *Proc. 8th Intl. Conf. on Embedded Software (EMSOFT'08)*, pages 69–78. ACM, 2008.