Learning-based Routing in Mobile Wireless Sensor Networks: Applying Formal Modeling and Analysis

Fatemeh Kazemeyni, Olaf Owe, Einar Broch Johnsen
University of Oslo, Norway
{fatemehk,olaf,einarj}@ifi.uio.no

Ilangko Balasingham
Oslo University Hospital, Norway
ilangkob@medisin.uio.no

Abstract

Limited energy supply is one of the main concerns when dealing with wireless sensor networks (WSNs). Therefore, routing protocols should be designed with the goal of being energy efficient. In this paper, we select a routing protocol which is capable of handling both centralized and decentralized routing. Mobility, a priori knowledge of the movement patterns of the nodes is exploited to select the best routing path, using a Bayesian learning algorithm. Generally, simulation-based tools cannot prove if a protocol works correctly, but formal modeling methods are able to validate that by searching for failures through all possible behaviors of network nodes. This paper presents a formal model for a learning-based routing protocol for WSNs, based on a Bayesian learning method, using an Structural Operational Semantics (SOS) style. We use the rewriting logic tool Maude to analyze the model. Our experimental results show that decentralized approach is twice as energy-efficient as the centralized scheme. It also outperforms the power-sensitive AODV (PS-AODV) routing protocol (i.e. a non-learning efficient protocol). We use the Maude tool to validate a correctness property of the routing protocol. Our formal model of Bayesian learning integrates a real dataset which forces the model to conform to the real data. This technique seems useful beyond the case study of this paper.

1 Introduction

Wireless sensor networks (WSNs) consist of sensor nodes which collaborate with each other to gather data and transmit the data to sink nodes. Sensor nodes usually have limited energy resources. They use routing protocols to find the path to transmit data to a designated sink node. Transmission is one of the most energy expensive activities in WSNs. So, routing protocols should be designed efficiently. The nodes store the total paths in a routing table. Having such routing tables is not always feasible, especially in large-scale ad-hoc networks. If nodes move frequently, the cost of finding paths increases dramatically. In addition, the rate of packet loss increases because of the frequent disconnection between the nodes. In these cases, learning methods could increase the system's performance. These methods predict the best action, by learning from previous actions.

Protocol evaluation is usually done through simulators, e.g. NS-2. Instead, we use formal techniques, which provide us with an abstract way to model protocols. Formal techniques can prove the protocol's correctness, by inspecting all reachable states of a system. Simulators cannot prove the correctness of protocol's properties, due to non-deterministic behaviors of WSNs (e.g. mobility and timing), but provide quantitative information about the protocol.

This paper introduces an approach to formally model the learning methods in WSN routing protocols. For this purpose, we aim to have a general model for learning-based routing protocols. This model is used to analyze and validate these kinds of protocols both quantitatively and formally. The motivation is to have a simplified model which includes the important features of the learning and adapting protocols and additionally the movement patterns of the nodes. We define routing protocols which estimate the most probable routing path for mobile WSNs, considering the Nodes' frequent movement. Each node's movement may have a pattern which is calculated, for every pair of nodes, as the probability of being connected. These probabilities are used to find the most probable existing path. We define a centralized and a decentralized approach for the routing protocol. In the centralized protocol, the routing paths are computed centrally by so-called *processing* nodes, using Dijkstra's Shortest Path Algorithm [5]. They have information of the whole network. Nodes request the best and most available routing path from the processing node. The notion of best is determined as a function of the cost of the links, i.e., communication links between the nodes of the network. In our model, these costs are determined based on the likelihood of availability and reliability of these edges. The best route is the one with the highest probability of successful message delivery. The centralized protocol is not very efficient because of the interactions between nodes and the

processing node(s). In the decentralized protocol, nodes are responsible for finding the path. We model a protocol in which each node learns which neighbor has the best transmission history by using the Bayesian learning method [14].

We design and develop a formal, executable model of our protocol as a transition system in a SOS style. This model is analyzed using Maude [4], i.e. a formal modeling tool based on rewriting logic [13]. To have a more realistic analysis of our model, we introduce the concept of facts in our model. Facts are deterministic actions that must occur according to prescheduled times, complementing the non-deterministic actions in the model. Probability distributions are applied to abstractly model the probabilistic behaviors of the network. We feed our model with the real data-set as a fact-base. By feeding a model with real mobile traces, we obtain a more reliable analysis, in addition to taking advantage of the formal analysis. Note that using facts does not restrict the actions that can be performed in the model, it only adds extra actions that are expected to happen. Our model is used to analyze both performance and correctness.

Related Work. There are a few work which consider frequent movements of nodes and its role in the successful transmission and the efficiency of networks. The PROPHET protocol [12] introduces the delivery predictability factor to choose the messages in networks which do not guarantee fully connected networks. Recently, reinforcement learning methods have been used in the network protocols to improve their efficiency and adaptability. Research in [20] and [16] propose WSN protocols which adaptively learn to predict optimal strategies to optimize their efficiency. There are centralized and also decentralized routing protocols. We have defined an adaptive and learning-based centralized and decentralized protocols, based on the main features of the energy-efficient and learning-based routing protocols. The most similar protocol to the protocol used in this paper is Q-probabilistic routing protocol [2] which uses Bayesian algorithm. This protocol achieves a trade-off between the energy usage and the number of retransmission. The main difference between our protocol and the Q-probabilistic protocol is that we assume that nodes are mobile and do not have any information about their location, whereas the Q-probabilistic protocol does not talk about mobility of nodes and assumes that each node knows its location and the distance to the destination.

Formal methods are much less explored to analyze the WSNs, but recently started to appear. Ölveczky and Thorvaldsen show that Maude is a well-suited modeling tool for WSNs [15]. Real-Time Maude is used to model and analyze the efficiency of the OGCD protocol [15]. We follow this line of research and use Maude to develop a learning-based routing protocol for WSNs. There are very few works which study learning techniques in formal methods. A reinforcement method which is used in a Q-routing protocol is modeled in [11], using SPIN model checker. In contrast to our work, this paper does not introduce a generalization for learning rules in formal models. The results in [11] are only qualitative, while we provide a realistic analysis both qualitatively and quantitatively, using a real data-set.

Paper Overview. Section 2 introduces reinforcement learning. Section 3 presents the generalized routing protocol which is modeled later in this paper. Section 4 presents the main contribution of the paper, which is integrating learning of observables in a probabilistic model. Section 5 discusses the modeling of the routing protocols, using the learning rules. Section 6 describes the case study and the analysis results. Section 7 concludes the paper.

2 Reinforcement Learning

Reinforcement learning is concerned with predicting the actions which maximize the accumulated rewards for agents [7]. Agents learn which actions to choose by observing the rewards obtained from previous actions. Figure 1 represents the interactions of an agent and the environment. The actions are chosen by agents and change the states. As a result, agents receive a *reinforcement learning signal* in the form of a reward. Reinforcement learning is formulated as a *Markov Decision Process*, which consists of a tuple (ST, A, P, R) where ST is a set of environment states, A is a set of actions, $P(st'|st, a)$ is the *probability* of being in state st' after choosing action a in state st, and $R(st, a, st')$ is the *reward* associated with this action.

Bayesian inference is an extension of reinforcement learning which updates the probability estimation of actions (called hypotheses), based on *Bayes* rules [3]. In Bayesian inference, a set of actions is defined and agents learn to estimate the probability of the actions and choose the action with the highest probability. Assume that there are k actions $A = \{a_1, a_2, ...a_k\}$, and D is the set of training actions (including the information which is obtained from the accumulated history of observations). $P(a)$ denotes the prior probability of action a (before any observation from the set of training actions), $P(D|a)$ is the probability of D given action a (indicating the perceived likelihood of observing the action which is actually observed), and $P(D)$ is the prior probability of the set training actions which is obtained from the previous actions ($P(D) = \sum_{a \in A} P(D|a)P(a)$). Then, $P(a|D)$ is the probability of action a given the set of actions D. The Bayesian method finds the maximum posteriori action a_{MAP}, which is the most probable action $a \in A$:
$a_{MAP} = argmax_{a \in A} P(a|D)$, which is equal to $argmax_{a \in A}(\frac{P(D|a) \times P(a)}{P(D)})$, and consequently

$$a_{MAP} = argmax_{a \in A}(P(D|a) \times P(a)), \qquad (1)$$

where, $argmax_{a \in A} P(a|D)$ returns the action $a \in A$ for which $P(a|D)$ attains its maximum value. Note that the

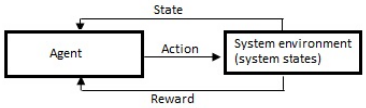

Figure 1. A learning flowchart.

action a which maximizes the value of $P(D|a) \times P(a)$, also maximizes the value of $\frac{P(D|a) \times P(a)}{P(D)}$. We use the Bayesian method for predicting the best action for routing.

3 The Selected Routing Protocol

This section explains the general learning-based routing protocols for WSNs, based on the node movement patterns and the probability associated to the communication links. Assume that we have a WSN with nodes which move frequently. We want to find the best routing path between the sensor and the sink. We consider the best path to be the one which is the most reliable and has the highest probability of availability during the message transmission. We assume that a WSN is a graph $G = (S, L)$, where $S = \{n_1, n_2, ..., n_N\}$ is the set of nodes and $L = \{l_{i,j} \mid \forall i, j \in S \text{ s.t. } n_i \text{ is within radio range of } n_j\}$ is the set of the communication links in G. We want to find the path that is most likely to deliver a message between two nodes. Each link in the graph has a weight, in terms of a set of probabilities which specify the effectiveness of the link in the message transmission. We define the links' weight as a combination of the in-range probability of the link and the reliability of the link. The link's reliability is important, as well as its availability. If a link is available but it is not reliable, messages may get lost. Each link $l_{i,j} \in L$ has a probability which we call the *in-range probability* or $EP_{l_{i,j}}$. This probability is calculated using the history of the availability of the network links. We note that if two nodes are not within radio range from each other, a link between them is not available, because nodes cannot receive each other's messages. Nodes meet when they enter the radio communication range of each other. Assume that an initial probability is defined for each link $l_{i,j}$, between two nodes i and j. The chance that two moving nodes meet each other (be in each other's range) during a period of time could increase or decrease the initial probability using the following definition similar to [19]:
$EP_{l_{i,j}} = EP_{l_{i,j}} \times (1 - f) + f$ if i and j meet within a predefined time interval T,
$EP_{l_{i,j}} = EP_{l_{i,j}} \times (1 - f)$ otherwise,
where f, the *adapting factor* $0 < f < 1$, is a predefined value which specifies the rate of probability update. The factor f, the initial probability of the links, and the time interval T, depend on the specific properties of a network. The specification can range from conservative, e.g., $EP_{l_{i,j}} = 0 \ \ for\ all\ i, j \in S$, to optimistic, e.g., $EP_{l_{i,j}} = N \times (\pi\omega^2/c) \ \ for\ all\ i, j \in S$, where ω and c are the radius of the radio range and the total area under study, respectively. Here, we assume all nodes are uniformly distributed within the area of study. The reliability of a link depends on different factors. For example, the physical nature of the environment between two nodes changes the reliability of their link in time. For simplicity, we consider the average historical reliability of the link $l_{i,j}$ as a probability $RP_{l_{i,j}}$, which is obtained from the network's history based on the observed reliability so far in the communication between the nodes i and j. The probability which is assigned as weight to each network link $l_{i,j}$, denoted as $P_{l_{i,j}}$, is the combination of these two independent probabilities: $RP_{l_{i,j}}$ and $EP_{l_{i,j}}$, namely $P_{l_{i,j}} = RP_{l_{i,j}} \times EP_{l_{i,j}}$.

When the total information of the routing paths is not available, specially in the decentralized approaches where the knowledge of each node for transmitting a message is limited to its neighbors, we use acknowledgment (ACK) messages to infer the probability of availability of the routing paths to each destination node. The source node s can calculate the acknowledgment probability $AP_{s,e,d}$ for the messages to the destination d through the neighbor node e, as $AP_{s,e,d} = \frac{ackNum^e_{s,d}}{msgNum^e_{s,d}}$, where $msgNum^e_{s,d}$ and $ackNum^e_{s,d}$ are the total number of the sent data messages from node s to d and the total number of the ACK messages from node d to s, respectively, in both cases through the neighbor node e. In the case of not having the total information of the routing path, we use the Bayesian learning method to predict which path has the highest chance to deliver a message successfully, based on the acknowledgment probability. Assume $M = \{e_1, e_2, ..., e_m\}$ is the set of neighbors of node s. According to the Bayesian learning method, there are m hypotheses $H = \{h_{e_1}, h_{e_2}, ..., h_{e_m}\}$, each refers to selecting neighbor node $e_k \in M$. We need to find h_{MAP} based on Formula 1. In our protocol, for each source node s and destination node d, $P(D|h_{e_k})$ is the rate of successful message transmission, if node s chooses neighbor node e_k to transmit the message toward d, which means $P(D|h_{e_k}) = AP_{s,e_k,d}$. and $P(h)$ is initially assigned to $P_{l_{s,e_k}} = RP_{l_{s,e_k}} \times EP_{l_{s,e_k}}$.

Each time that node s wants to send a message, it calculates $P(D|h_{e_k})$ for all neighbors $e_k \in M$ and chooses the hypothesis (neighbor node) with the highest probability. If the message is transmitted successfully, an acknowledgment is sent from d to the source node s. By receiving this message, s is rewarded by updating the reward table ($ackNum^e_{s,d}$). This reward affects the next predictions by improving $AP_{s,e,d}$. This ratio can give us an estimate about the probability of the availability of a path. The accuracy of this estimation improves during the time, by observing more transmissions. Consequently, this ratio shows the chance of a successful transmission through that path. We assume that the learning time is less than the movement time to avoid possible learning failures (e.g. missing a movement).

To compare centralized and decentralized learning, we define two versions of this protocol, one centralized and another, more robust and flexible, which is decentralized. There is a trade-off between using these approaches. In the centralized approach, central nodes needs to have information about the total system. Gathering and storing the information are costly, and sometimes impossible. The decentralized approaches are often less accurate, but more efficient. When processing nodes are not reachable, the decentralized protocol is the only feasible option.

3.1 The Centralized Approach

In the centralized approach, we use a modified Dijkstra's algorithm to find the most probable path. This modification returns all the nodes in the best path, while Dijkstra's algorithm only returns the best path's weight. To calculate a link's weight, we use the links' probability, instead of the nodes' distance which is used by the original Dijkstra algorithm. This is a centralized process which is performed in the processing node. When sensor nodes move, they inform the processing node about their new position. Each time a node needs a path, it sends a message to the processing node and asks for the path. The processing node sends a reply message to the node which includes the result path.

3.2 The Decentralized Approach

In the decentralized protocol, each node decides locally which node should be chosen as the next node in the routing path. Nodes only have the information of themselves and their neighbors, not the total information of the network. There are some methods which can be used in these situations, such as the "ant colony" [6]. But these methods frequently rebroadcast messages asking for routing paths, which makes them less energy-efficient. Instead, we let each node collect information from previous transmissions to learn how to predict the best route. We note that nodes do not have all the information ideally needed to decide about the total path, but use what is available to them, to pass a message to the node about which path to the destination is the most probable. In the decentralized approach, each node uses the Bayesian learning method to predict the neighbor with the highest chance of successful message delivery.

4 Integrating Learning of observables in a probabilistic model of WSNs

This section integrates the reinforcement learning aspects discussed in the previous section with a probabilistic WSN model, presented in SOS style [17]:

$$\frac{Conditions\ ,\ \ sample(\pi)}{conf \longrightarrow\ conf'} \quad \text{(RULE NAME)}$$

This rule locally transforms configurations or subconfigurations which match *conf* into $conf'$ if the $Conditions$ hold and with a sample from a Boolean valued probability distribution π. Configurations consist of nodes, messages and time information. Nodes have the form $obj(o, n, rt)$, where o is the node's identifier, n is the list of neighbors' information and rt is the reward table. Messages have the form $msg(m, s, d)$, where m is the content of the message (the name and the parameters of the message), and s and d are the source and the destination nodes. In order to handle scheduling of the nodes, a discrete *time* is added to the global configuration of the system, and *execution marks* and *scheduled execution marks* sorts are added as subsorts of Configuration. The global time has the form $time(t)$, where t has a floating point value. Execution marks have the form $exec(o)$ and scheduled execution marks have the form $[t, exec(o)]$ where o is the node's identifier and t is the scheduled time. The auxiliary function sampleBe(δ) samples the Bernoulli distribution with rate δ and sampleExp(δ) samples an exponential distribution with rate δ, and are assumed to be predefined. The sampleBe(δ) function returns a Boolean value and the sampleExp(δ) function returns a float value, by considering probability distribution with rate δ (Their definition can be found in textbooks, e.g., [8]).

4.0.1 Generic Execution Rules for Enabled Transitions

We use a ticketing system to obtain a probabilistic scheduling of enabled transitions, following an approach used with PMaude [1]. The idea is that a node needs a ticket to execute, after which its ticket is delayed for some time determined by an exponential probabilistic distribution. The probabilistic scheduling works in combination with time advance: time cannot advance when a node has an enabled ticket, and time can only advance until the next scheduled time/ticket in the configuration. Object execution is captured by a set of rules with the general form of

$$\frac{Conditions}{exec(o)\ obj(o, ...) \longrightarrow\ done(o)\ obj(o, ...)} \quad \text{(RULE NAME)}$$

where the left-hand and/or the right-hand side may additionally involve messages, and the global time. Here, $done(o)$ indicates that node o is executed and can be rescheduled. The scheduling is done in the *Probabilistic ticketing* rule. The *Tick* rule changes the time, considering the next execution ticket. The tickets are enabled one by one, using the *Enabling* rule. In the *tick* rule, δ is a predefined value and $maxTimeAdv(conf)$ returns the maximum possible time advance (i.e. the minimum scheduled time).

$$\frac{t' = t + sampleExp(\delta)}{done(o)\ time(t) \longrightarrow\ [t', exec(o)]\ time(t)} \quad \text{(PROBABILISTIC TICKETING)}$$

$$\frac{t' = maxTimeAdv(conf),\ t' > t}{\{conf\ time(t)\} \longrightarrow\ \{conf\ time(t')\}} \quad \text{(TICK)}$$

(ENABLING)

$$\frac{}{[t, exec(o)]\ time(t) \longrightarrow exec(o)\ time(t)}$$

4.0.2 The Rule for Lossy Links

Links may be lossy. It means that in the model, each link has an associated probability which reflects the link's reliability. Therefore, the chance of losing messages sent between nodes *o* and *o'* depends on the link between *o* and *o'*. The message loss is captured by the *Lossy Link* rules. Messages are lost if $sampleBe(\delta)$ is true (*Lossy Link 1* rule), otherwise they remain untouched (*Lossy Link 2* rule). In these rules, *send(M,o,o')* represents messages not yet delivered, and *linkProbability(o,o',conf)* is the probability reflecting the reliability of the link between *o* and *o'*.

(LOSSY LINK 1)

$$\frac{\delta = linkProbability(o, o', conf),\ sampleBe(\delta)}{send(M, o, o')\ conf \longrightarrow msg(M, o, o')\ conf}$$

(LOSSY LINK 2)

$$\frac{\delta = linkProbability(o, o', conf),\ \neg sampleBe(\delta)}{send(M, o, o')\ conf \longrightarrow conf}$$

4.1 The Rule for Reinforcement Learning

Regarding the learning theory (Figure 1), we define three categories of rules to model a learning process.

4.1.1 The Rule for Forcing Observed Facts

In contrast to non-deterministic configuration changes, there are some deterministic actions in the system (we call them *facts*), that must be forced to perform according to prescheduled points in time. In this model, facts are specified in a fact-base FB. These facts represent actions that are observed from the environment (e.g. the movement of nodes in a real network). So, we schedule them to happen in the model as they happen in reality. These actions should be performed at the prescheduled times. Actions in FB have the general form of $action(o, data, t)$, where $data$ is defined based on the system which is under study. In this case study, actions in FB are defined as node movements which cause a node to meet new nodes. Here, $data$ represents the neighbors which node o meets at time t. In the *Facts* rule, node o modifies its neighbor list n when receives an action message, by appending the new neighbors mentioned in $data$ to n. By including FB in the configuration, the *maxTimeAdv(conf)* function (in the *Tick* rule) considers the smallest time of unexecuted actions in FB as well as the nodes' execution tickets to choose the maximum possible time advance. Thereby we avoid missing any action in FB.

(FACTS)

$$\frac{FB' = FB \backslash \{action(o, data, t)\}\quad action(o, data, t) \in FB,\ n' = append(n, data)}{obj(o, n, rt)\ time(t)\ FB \longrightarrow obj(o, n', rt)\ time(t)\ FB'}$$

4.1.2 The Rule for Predicting Actions

When node s wants to send data to node d, it generates a message. This message is transmitted through a path of nodes leading to d. Each time node o (i.e. a node in the path) transmits a message (with source node s and destination d), it needs to predict which neighbor $o" \in N$ has the best chance to pass the message successfully. This prediction is based on the Formula 1 and by having the previous rewards' information (in the reward table rt) which is updated by the rewards that were obtained from the previous successful transmissions (acknowledgment messages). The reward table rt has a row corresponding to the delivery rate to destination d if choosing neighbor $o"$. It is stored by the pair (s, a) which contains the number of successful transmissions and the total number of transmissions.

(PREDICT)

$$\frac{o" = prediction(n, rt, d), rt(o", d) = (s, a), rt' = rt[(o", d) \mapsto (s, a + 1)]}{msg(data(s, d, path), o', o)\ exec(o)\ obj(o, n, rt) \longrightarrow send(data(s, d, (path; o")), o, o")\ done(o)\ obj(o, n, rt')}$$

4.1.3 The Rule for Rewards

Each node is rewarded after a successful action (receiving an acknowledgment message). The reward table of nodes stores the information of the message transmissions, captured by the *Reward* rule.

(REWARD)

$$\frac{o' = next(o, path)\ rt(o', d) = (s, a),\ rt' = rt[(o', d) \mapsto (s + 1, a)]}{msg(ack(path), d, o)\ exec(o)\ obj(o, n, rt) \longrightarrow done(o)\ obj(o, n, rt')}$$

The application of these general rules are shown in the model of the centralized and the decentralized protocols.

5 Application of the Learning Rules

We have applied the learning rules in a case study, which includes the generalized learning-based routing protocol of Section 3. The case study is modeled in rewriting logic, using the Maude tool. In this section, we present SOS style rules for the decentralized protocol which reflects the learning process in a more abstract way. A complete set of Maude rules of both the centralized and the decentralized protocol is presented in [10]. In this model nodes have the form $\mathbf{node}(o, e, n, rt)$. Here, o is the node's identifier, e is a pair of $(power, energy)$, where $power$ is the power capability and $energy$ is the total remaining energy in the node, n is the node's neighbors information, and rt is the reward table. The execution rules for enabled transitions, the rules for lossy links and the *Facts* rule can be used in this case study as they are presented in Section 4. The $Predict$ and $Reward$ rules depend on the protocol and need some modifications to reflect the details and requirements of different protocols. According to the requirements of this case study, the prediction rule is split into three rules. The $Predict1$

rule performs when the destination node is in the neighborhood. The second rule is related to the situation that the destination is not in the neighborhood and the node uses the learning method to choose a node to pass the message. In the $Predict3$ rule, there is no neighbor to transmit the message. Each unicast message transmission is sent by consuming the minimum message transmission power of the nodes. When the destination node receives a data message, it broadcasts an ack message to all the nodes in the routing path, using its maximum transmission power (the *Acknowledgment rule*). The $Reward$ rule updates the reward table of nodes after receiving an ack message.

$$\frac{d \in n,\ rt(d,d) = (s,a),\ rt' = rt[(d,d) \mapsto (s,a+1)] \quad e' = (power, energy - power^{min})}{msg(data(id,s,d,path),o',o)\ exec(o)\ node(o,e,n,rt) \longrightarrow send(data(id,s,d,(path;o)),o,d)\ done(o)\ node(o,e',n,rt')} \text{(PREDICT1)}$$

$$\frac{n\backslash\{o'\} \neq nil,\ d \notin n, o" = prediction(n,rt,d,o,o',t,0,0) \quad rt(o",d) = (s,a),\ rt' = rt[(o",d) \mapsto (s,a+1)] \quad e' = (power, energy - power^{min})}{msg(data(id,s,d,path),o',o)\ exec(o)\ node(o,e,n,rt) \longrightarrow send(data(id,s,d,(path;o)),o,o")\ done(o)\ node(o,e',n,rt')} \text{(PREDICT2)}$$

$$\frac{n\backslash\{o'\} = nil}{msg(data(id,s,d,path),o',o)\ exec(o)\ node(o,e,n,rt) \longrightarrow done(o)\ node(o,e,n,rt)} \text{(PREDICT3)}$$

$$\frac{e' = (power, energy - power^{max})}{msg(data(id,s,d,path),o,d)\ exec(d)\ node(d,e,n,rt) \longrightarrow send(ack(id),d,path)\ done(d)\ node(d,e',n,rt)} \text{(ACKNOWLEDGEMENT)}$$

$$\frac{o \in path,\ o' = next(o,path), e' = (power, energy - power^{min}) \quad rt(o',d) = (s,a), rt' = rt[(o',d) \mapsto (s+1,a)]}{msg(ack(id),d,path)\ exec(o)\ node(o,e,n,rt) \longrightarrow done(o)\ node(o,e',n,rt')} \text{(REWARD)}$$

The $Predict2$ rule uses the $prediction$ function. It is a recursive function which chooses one of the node's neighbors with the best estimation regarding the related link reliability(lr) and in-range probability(ep) and the accumulated rewards in the reward table regarding each neighbor. It also calls the $calcRew$ function which calculates the stored rewards in the reward table for each neighbor node. The ; symbol connects an element or a sublist to the list. Functions *firstNode(rt), secondNode(rt), numOfRecMsg(rt)* and *numOfSentMsg(rt)* return the node's neighbor, the destination, the number of messages sent through this neighbor and the number of these messages received by the destination.

function prediction$((n_1;n_2),rt,o,o',o'',t,i,f)$ =
  if $f' > f$ **and** $t <$ time(n_1)
  then prediction$(n_2,rt,o,o',o'',t,nodeID(n_1),f')$
  else prediction(n_2,rt,o,o',o'',t,i,f)
  where f'=(calcRew$(rt,i,o)\times$(lr$(n_1)\times$ep$(n_1))$).
function calcRew$((rt_1;rt_2),o_1,o_2)$=
  if firstNode$(rt_1) = o_2$ **and** secondNode$(rt_1) = o_2$
  then (numOfRecMsg(rt_1)/numOfSentMsg(rt_1))
  else calcRew(rt_2,o_1,o_2).

6 Analysis

Formal tools allow us to analyze the learning-based model both quantitatively and qualitatively, in contrast to simulators which can only do quantitative analysis. Although our WSN formal models are quantitatively analyzed, our results are not directly comparable with the simulators' results, because formal models provide an abstract presentation of WSNs (i.e. the details considered by simulators, such as SINR, are not included in formal models). We used the dataset of iMotes devices in the Intel Research Lab [18] to evaluate the models. An iMote is a small and self-contained device, which is able to communicate with the other iMotes through radio links. This dataset represents Bluetooth sightings of 16 moving iMotes. Each row of the dataset includes a pair of these iMote nodes which meet (are within each other's radio communication range), in addition to the starting time and the length of this meeting. We assigned the same value of 3/4 to all the links between all the nodes. That is, we assume that the links are fairly reliable, which is a reasonable assumption, otherwise the very attempt to use such a WSN should be questioned. Note that as mentioned in Section 3, links may be available but not reliable (e.g. because of environmental conditions such as foggy weather). So, a message between neighbors may still be lost. We assume that nodes use a given transmission power to reach their neighbors (the node's minimum power), and this transmission power is two times larger than the cost of receiving a message; broadcasting is done with the maximum power. Using such power ratio frees our experiment from any particular radio model. We assume all nodes can reach a processing node and nodes (except the stationary one) move according to the dataset's specification. Every node has an initial energy budget of 1000 units. We run experiments to investigate the protocol's efficiency, effectiveness and some of its formal properties.

6.1 Investigating the Efficiency and Delivery Rate

In the first experiment, each node sends one single data message to the sink. The purpose of this experiment is to study the overhead cost of the learning phase in these protocols, as well as the cost incurred by the nodes in the centralized approach in updating the processing node every time they move. Figure 2 compares the average remaining energy of all the nodes which send the data message after 3000 time ticks. The figure shows that the decentralized protocol consumes nearly 1/3 less energy than the centralized protocol at the end of simulation run. Since there is one message being delivered, we claim that the process of keeping the processing node updated is to be blamed as a non-trivial consumer of energy. One could argue that we should have compared our protocols with a non-learning protocol, such

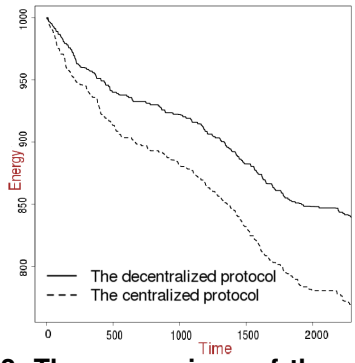

Figure 2. The comparison of the average energy consumption of the centralized and the decentralized protocols for one data message.

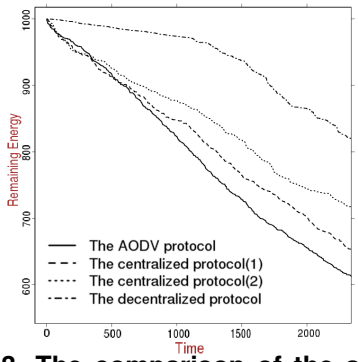

Figure 3. The comparison of the average remaining energy of nodes in the decentralized, the centralized protocols (1) and (2).

as the power-sensitive AODV (PS-AODV) [9], i.e., a modified version of AODV which finds the cheapest routing path instead of the shortest one. We argue that we would not learn anything because the PS-AODV does not have a learning phase and since there is only one message being sent, therefore this comparison would not be realistic. We did another experiment to study the power consumption of the network when all the nodes regularly send data messages. This experiment includes a comparison to PS-AODV. In this experiment, each node sends 50 data messages at random times. Figure 3 represents the average of the remaining energy of all nodes during running the decentralized, the centralized and the PS-AODV routing protocol models. Again, each simulation runs for 3000 time ticks, and we assume there is no message interference. This experiment, allows one to better appreciate the cost of the route requests in the centralized protocol (in addition to the cost of maintaining the processing node up-to-date). In PS-AODV, we assume that the links always exist. Note that those are rather *optimistic* assumptions. In PS-AODV, nodes need to request a new path by broadcasting route request messages. We assume that nodes also use their minimum power to broadcast the messages. As before, the centralized and the decentralized protocols spend some time in the beginning for learning. We also investigated two scenarios for the centralized protocol. In the first scenario (denoted as "centralized protocol (1)"), we assume, that communicating with the processing node is as costly as communicating with a neighbor. The second scenario (denoted as "centralized protocol (2)") has similar situation as the first scenario except that nodes use twice as much power to communicate with the processing node. While this improves the chances that a processing node is reached, it also consumes more energy. The results show that in the long term the decentralized protocol consumes 45.9%, 53.5% and 59.7% less power than the first and second scenarios of the centralized protocol and PS-AODV, respectively. We note that (*i*) the centralized and the decentralized protocols consume less energy than PS-AODV, and (*ii*) the decrease in the average amount of remaining energy is relatively less pronounced for the decentralized protocol, which is a desirable feature as well. In the other experiment, all nodes continue moving and sending messages at random intervals until the first node dies. In the both protocols, we show the minimum amount of energy among all nodes. Here we compare the centralized protocol(1) and the decentralized protocol. Figure 4 shows that the first node dies in the centralized protocol after approximately 5000 time ticks, while the most deplete node in the decentralized protocol still has almost half of its initial energy budget. More efficiency analysis can be found in [10]. Considering the successful transmission rate as the metric of interest, the centralized protocol, as expected, is more effective than the decentralized. On average, the successful message transmission rate of the centralized protocol is 87.1%, while it is 83.6% for the decentralized protocol. According to the average power consumption in Figure 3, the decentralized protocol uses nearly 50% less power than the centralized protocol at the cost of being merely about 4% less effective. This is quite acceptable trade-off. That is, the decentralized protocol revealed itself as a good alternative to the centralized one, as well to PS-AODV.

6.2 Investigating the Formal Properties

We took the advantage of the formal modeling methods to prove the validity of the model, according to a correctness property. The correctness of the routing protocols is formalized as a *Linear Temporal Logic (LTL)* formula $\Box(MSG(m,s,d) \Rightarrow MPP(m,s,d))$.

The predicate $MSG(m,s,d)$ is true if a message m is sent from node s to node d. The predicate $MPP(m,s,d)$ is true if the chosen path for m is the most probable path between nodes s and d. This formula means that in all states of the system, the most probable paths are chosen. Note that it does not guarantee the delivery of messages (due to the unreliability of links). MPP is defined separately

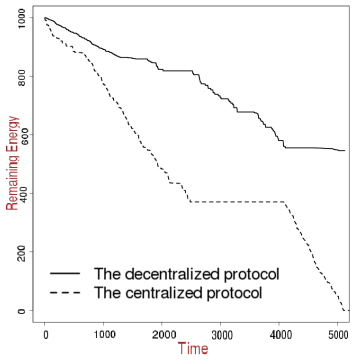

Figure 4. The comparison of minimum remaining energy of nodes in the decentralized and the centralized protocol.

for the two versions of protocol. The centralized protocol uses available global information and the decentralized one uses available local information to calculate the value of MPP. Maude has a *search* tool that searches for failures (the negation of the correctness property). We used this tool to check the correctness property by searching all the reachable states of the system to find out if the most probable path is always chosen. For this experiment, nodes only send one message to have a limited search state space. The search tool did not find any violation of the property, meaning that the expected path is chosen in all the reachable traces of the model's run, thus asserting the correctness of the protocol.

7 Conclusion

In this paper, we proposed a model for learning-based protocols. We integrated the rules required for learning the observables with a probabilistic model of WSNs. The model includes the concept of a fact-base, which consists of the scheduled actions that happen in certain orders. We have defined a routing protocol which includes the Bayesian learning method to predict the best routing path. This protocol has centralized and decentralized versions. We used Maude to model the routing protocol and to formalize the Bayesian learning method. Besides applying Maude facilities to analyze the model, we provided more realistic analysis by feeding the model by a real dataset and managing quantitative data analysis. The results show a trade-off between two versions of the protocol. The centralized one is slightly more effective (higher delivery rate), while the decentralized one is much more efficient. Both versions of the protocol outperform the PS-AODV protocol in term of efficiency and are also more robust, especially the decentralized one. We have qualitatively analyzed the protocol and proved that the protocol satisfies its correctness property.

In future work, we are going to refine the protocol by capturing more detailed patterns from the node movements, e.g., temporal patterns. We also plan to perform probabilistic reasoning of the model via statistical model checking.

References

[1] G. Agha, J. Meseguer, and K. Sen. PMaude: Rewrite-based Specification Language for Probabilistic Object Systems. *ENTCS*, 153(2):213–239, 2006.

[2] R. Arroyo-Valles, R. Alaiz-Rodriguez, A. Guerrero-Curieses, and J. Cid-Sueiro. Q-probabilistic routing in wireless sensor networks. In *the 3rd Intl. Conf. on Intelligent Sensors, Sensor Netw. & Info.*, pp.1–6,2007.

[3] G. E. P. Box and G. C. Tiao. *Bayesian Inference in Statistical Analysis* . Wiley-Interscience, 1992.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: Specification and programming in rewriting logic. *Theor. Comp. Sci.*, 285:187–243, 2002.

[5] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[6] M. Dorigo and T. Stutzle. *Ant colony optimization*. MIT Press, Cambridge, 2004.

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *In JAIR*, 4:237–285, 1996.

[8] J. Kalbfleisch. *Probability and Statistical Inference*. Number v.1 in Springer texts in statistics. 1985.

[9] F. Kazemeyni, E. B. Johnsen, O. Owe, and I. Balasingham. Formal modeling and validation of a power-efficient grouping protocol for wsns. *JLAP*,81(3):284–297,2012.

[10] F. Kazemeyni, O. Owe, E. B. Johnsen, and I. Balasingham. Learning-based Routing in Mobile Wireless Sensor Networks: Formal Modeling and Analysis for WSNs. *Technical Report:ISBN 82-7368-390-7* Uni. of Oslo,2013.

[11] S. A. Kulkarni and G. R. Rao. Formal modeling of reinforcement learning algorithms applied for mobile ad hoc network. *In IJRTE*, 2:43–47, 2009.

[12] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mobile Comp. & Comm.*,7:19–20,2003.

[13] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theor. Comp. Sci.*, 96:73–155, 1992.

[14] T. Mitchell. *Machine Learning*. McGraw-Hill,1997.

[15] P. C. Ölveczky and S. Thorvaldsen. Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude. *Theor. Comp. Sci.*, 410(2-3):254–280, 2009.

[16] C. Pandana and K. J. R. Liu. Near-optimal reinforcement learning framework for energy-aware sensor communications. *In JSAC* ,23:209–232, 2002.

[17] G. D. Plotkin. A structural approach to operational semantics. *In JLAP*, 60-61:17–139, 2004.

[18] J. Scott, R. Gass, J. Crowcroft, P.Hui, C.Diot, and A.Chaintreau. cambridge/haggle/imote/intel(v.2006-01-31).

[19] S. Shakya, J. McCall, and D. Brown. Using a markov network model in a univariate eda: an empirical cost-benefit analysis. *In GECCO '05* , pp.727–734, 2005.

[20] P. Wang and T. Wang. Adaptive routing for sensor networks using reinforcement learning. *In the 6th IEEE Conf. on Comp. & Info. Tech.* , pp.219–219.2006.