Journal of Universal Computer Science, vol. 20, no. 9 (2014), 1373-1401
submitted: 26/7/13, accepted: 16/6/14, appeared: 1/9/14 © J.UCS

Formal Study of Routing Protocols for Wireless Sensor Networks

José Antonio Mateo
(Universidad de Castilla La-Mancha, Spain
JoseAntonio.Mateo@uclm.es)

María del Carmen Ruiz
(Universidad de Castilla La-Mancha, Albacete, Spain
MCarmen.Ruiz@uclm.es)

Hermenegilda Maciá
(Universidad de Castilla La-Mancha, Albacete, Spain
Hermenegilda.Macia@uclm.es)

Juan José Pardo
(Universidad de Castilla La-Mancha, Albacete, Spain
Juanjose.Pardo@uclm.es)

Abstract: NORA (Network rOle-based Routing Algorithm) and NORIA (Network rOle-based Routing Intelligent Algorithm) are novel routing algorithms for Wireless Sensor Networks (WSNs), which combine various effective techniques in order to reduce energy consumption and improve data routes. NORA is an algorithm, which uses local and neighbourhood information to assign a role to each node on the net, whereas NORIA adds a fuzzy logic engine to NORA in order to improve this assignment. These algorithms are far from being trivial, and, therefore, there is a clear need for the use of formal methods to check their correctness and performance, prior to their deployment in a real environment. To this end, this paper presents a neat and rigorous study of both algorithms, and, for the sake of completeness, we study and compare also both with a well-known routing protocol: Tree Routing. Finally, Coloured Petri Nets (CPNs) have been chosen as an appropriate modelling language, using the well-known tool, CPNTools, to conduct our experiments.

Keywords: Wireless Sensor Networks, Performance Evaluation, Coloured Petri nets, Routing Algorithms.

Categories: D.2.4 Software/Program Verification, D.2.10 Design, F.1.1 Models of computation, F.4.3 Formal Languages

1 Introduction

A wireless sensor network is a network of many spatially distributed resource-constrained devices (nodes) which collect information and forward it through the network to a main location, usually named base station (or sink). To this end, the nodes might run previously an algorithm to discover their direct neighbours as well as to find routes to reach nodes not directly connected to them. This process is called *network discovery.*

We are dealing here with proactive multi-hop routing protocols. A proactive protocol sets up routing paths and states before there is a demand for routing traffic. Paths are maintained even if there is no traffic flow at that time. Usually, multi-hop routing will consume less energy than direct communication. However, multi-hop routing introduces significant overhead for topology management and medium access control.

Furthermore, wireless sensor networks pose some challenges that designers must take into account such as short lifetime since the nodes are highly energy constrained, unreliable communication due to the presence of a wireless medium and a clear need for self-configuration, requiring little or no human intervention. However, several unique features exist in wireless sensor networks that do not exist in general adhoc networks.

Since sensors are often deployed in remote applications like at the bottom of an ocean, in a biologically or chemically contaminated field, attached to animals, in forest area, the battery cannot be replaced frequently due to inaccessibility of sensor nodes. To prolong network lifetime, energy spending should be minimum so that, the efficiency of sensor networks strongly depends on the routing protocol used. Thus, routing protocols require generally two phases: setup and maintenance phase. In setup phase, protocols basically create the routes between the nodes in the network, whereas, in the maintenance phase, the validity of these routes is checked and/or updated. We focus here on the setup phase although the mechanisms used by NORA and NORIA in the maintenance phase can be included in our models. Nevertheless, when using formal methods in the specification of systems, some level of abstraction is required in order to make the analysis feasible. Therefore, we abstract away in this paper the mobility of nodes as well as other intrinsic characteristics of sensor networks such as potential node failures or new nodes joining the network. In Section 5, we describe how this is controlled in NORA (NORIA uses the same approach) and how it can be included in our Petri nets model.

In this setting, tests and simulations are usually carried out by designers to analyse the algorithms before the deployment of the nodes in a real scenario. Although these techniques give us an excellent overview of the protocol behaviour, they suffer from significant problems that one cannot ignore. First, simulation results are highly subordinated to the simulator used in the experiments, and, therefore, the results obtained are hardly transferable to other tools. Second, the underlying theory is often unclear or inaccessible to users. Finally, some undesirable aspects could still be undiscovered by using simulation, such as deadlocks or livelocks. Thus, we advocate for the use of both approaches, that is, one can use a formal model to analyse the protocol up to the analysis becomes infeasible, and, then, use simulation (when it is possible) for bigger scenarios. We opted here for this approach.

Before we begin, it is worth to mention that our work does not attempt to demonstrate the suitability of formal methods for the analysis of WSNs protocols. We present a real work which has been conducted in collaboration with the designers of the protocols who know the importance of studying new protocols in the early stages of design as it saves time and money among the most important. Moreover, it is obvious that in a field such as WSNs, where in many cases there are no standards and new protocols emerge almost daily, it is particularly necessary to have a method to assist in the early stages of design as well as to evaluate the flood of new protocols in order to compare their behaviour objectively.

In this paper, we carry out a rigorous comparison between two recent algorithms, NORA [Ortiz 2011] and NORIA [Ortiz et al. 2011], and another one well known, Tree Routing (TR). All of them are designed for the same purpose, establishment of sensor networks, but each works differently. We will present a deep and neat study of these algorithms which are been modelled using Petri Nets. Once the protocols have been presented, we will make a comparison between TR and NORA, due to the latter emerged as an alternative to the first. We will check what has been improved. NORA protocol was improved by adding fuzzy logic, resulting in NORIA protocol which has been compared with NORA to evaluate if this extension really brings significant improvements.

The rest of the paper is organised as follows. Next section gives an overview of related work. Later, Prioritised-Timed Coloured Petri Nets (PTCPNs) are introduced. In Sections 4, 5 and 6 the protocols under study (TR, NORA and NORIA) are described and modelled by PTCPNs. In Section 7, we compare the analysis and simulation results obtained in CPNTools for the protocols and, finally, in Section 8, our conclusions and future work are summarised.

2 Related Work

Due to the interesting challenges presented previously for wireless networks evaluation, several methods have been used to study various aspects of wireless networks. These studies that have been mostly done with simulation-based tools, using NS-2, OMNeT++, and extensions such as Castalia [Pham et al. 2007] and SensorSim [Park et al. 2000], although they have significant problems (as mentioned in the previous section).

Since this paper studies two recent algorithms, it is difficult to find related works to compare them directly with this work. Thus, we present in this section a bunch of works that use formal methods to model and analyse protocols. Most of these works are focused on modelling and validation of routing protocols for mobile ad-hoc networks (MANETs).

The works that come closest to the study that we performed in this paper are found in two very close works which present the CPN modelling of the Dynamic MANET On-demand (DYMO) routing protocol. Billington et al. in [Billington et al. 2009] use simulation to study properties of the protocol. Espensen et al. in [Espensen et al. 2008] present some initial state space analysis results. Besides, in the work of Espensen et al., we have found an idea that will serve as a major improvement in our work and we intend to do as future work: organising the CPN model into modules to reduce the complexity and make easier possible changes or improvements the protocol may suffer in the future.

When we focus on sensor networks, we find that the research field is still very wide. We study routing algorithms with a special interest in quantitative aspects, mainly in temporal and energy consumption properties. We have found a work in [Yue et al. 2010] that has attracted our interest because it studies the energy consumption although on a different network layer (MAC layer) than we work (network layer). They analyse energy consumption for gMAC protocol using the discrete-event simulator from the Mbius tool suite. The protocol includes a fully decentralised slot allocation algorithm. For static and simple mobility scenarios (rotating a single fixed row in the grid by one position), different grid positions of nodes sending initially, and a simple interference model with fixed communication range are assumed (not a realistic wireless model). The mobility results are limited. Next to this idea, we have found the work presented in [Yue and Katoen 2010b] where energy consumption for a randomised leader election protocol is studied. They consider a Markovian Decision Process model for interference (consisting of two states only), which they analyse using PRISM. Analysis results show that prioritising stations with higher power level reduces the overall energy consumption, it also changes the respective stations probability of being elected leader.

In short, there are a large number of formal studies in wireless networks although the most significant studies have been carried out on wireless networks are mainly focused on MANETs, where the special restrictions of sensor networks are not taken into account. Recently, there have been proposed new formal studies on WSNs although, as happened with the previous ones, they are limited to study existing protocols with the ultimate aim of undertaking a study. We believe that our work should go a little further and that the use of formal methods should collaborate in the design of these protocols. We were also very surprised that there are almost no studies focusing on something vital in WSN as energy consumption given that the network life depends entirely on this factor. To the best of our knowledge, the first formal approach for modelling and analysing of a Role-Based Routing Algorithm for Wireless Sensor Networks in which real networks have been taken into consideration was our work [Ruiz et al. 2012] where NORA protocol was study. This work was of great help for the protocol designers. So, when they considered to improve it by adding fuzzy logic, we continue lending our collaboration in the early stages of design, resulting in NORIA which is also presented in this work. On the other hand, we also believe that this work is a significant improvement with those found in the literature. Our formal analysis is limited to five nodes because we have the well-known problem of state explosion that presents the other jobs (although some studies limit their study to two or three states) but we can get to analyse networks of up to 177 nodes using simulation made with CPN Tools.

3 Prioritised-timed coloured Petri nets

In this work, we advocate the use of Prioritised-Timed Coloured Petri Nets (PTCPNs) [Jensen and Kristensen 2009], which are an extension of Petri nets. Petri nets are a mature formalism, based on a rigorous and sound theoretical background, that offers designers a clear and neat graphical representation, and a lot of extensions such as colours, time, priority and so on, improving the expressiveness of them. In addition to this, one can find a diverse range of tools for constructing and evaluating CPN models. We opted for CPNTools [CPNtool 2013], since it is the considered de-facto standard tool for editing, simulating and analysing coloured Petri nets, and, moreover, it supports hierarchical nets, priorities and time. This tool exhibits also two approaches to verify the correctness of a model: *formal verification* by analysing the state space of the model, and providing also the necessary machinery to make queries (written as CPN ML functions), and, *simulation*, which permits to obtain interesting conclusions about the system under review without making an exhaustive search in its state space.

In PTCPNs, places have an associated colour set (data types). Each token has then an attached data value (*token colour*), which belongs to the colour to which the token is associated. We will use timed colours, for which the first component will be a non-negative integer value, representing the data value, and the second component will be the token timestamp, a natural number representing the time at which the token will be available.

There is also a discrete global clock that represents the total time elapsed in the system model. Moreover, arcs have also an associated inscription (*arc expressions*), constructed using variables, constants, operators and functions. To evaluate an arc expression we need to bind the variables that are part of the expression with their current value, that is, this binding consists of assigning a value to the variables that appear in the arc inscription. These values are then used to select the token colours that must be removed or added when firing the corresponding transition.

Arc expressions can also have associated time information both for place-transition and transition-place arcs. However, only time inscriptions are needed in output arcs, and even, when all the output arcs of a transition have the same time inscription, there is a shorthand notation in CPNTools by which this time information is associated with the transition instead of the output arcs. The time inscription associated with a transition is used to specify the delay that must be added to the current value of the global clock for every token generated by the firing of the transition.

Transitions can also have associated guards, which are Boolean expressions that can prevent their firing. Thus, when a transition has a guard, it must evaluate to true for the binding to be enabled, otherwise the binding is disabled and the transition cannot be fired. Next, we define the specific model of prioritised-timed coloured Petri net [Jensen and Kristensen 2009] considered here.

Definition 1 (Prioritised-timed coloured Petri net). A Priotitised-Timed Coloured Petri Net is a 10-tuple CPN_T = (P, T, A, Σ ,V,C, G, E, I,Π) where:

- P is a finite set of places.
- T is a finite set of transitions such that $P \cap T = \emptyset$.
- $A \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs.
- Σ is a finite set of non-empty colour sets. Each colour set is either untimed or timed.
- V is a finite set of typed variables such that $\forall v \in V$ *Type[v]* $\in \Sigma$.
- $C : P \rightarrow \Sigma$ is a colour set function that assigns a colour set to each place. A place p is timed if C(p) is timed, otherwise p is untimed.
- $G : T \rightarrow EXPR_V$ is a guard function that assigns a guard to each transition t, which is required to be a boolean expression, i.e. $Type[G(t)] = Bool$.
- $E : A \rightarrow EXPR_V$ is an arc expression function that assigns an arc expression to each arc a such that
  - *Type*$[E(a)] = C(p)_{MS}$ if p is untimed;
  - *Type*$[E(a)] = C(p)_{TMS}$ if p is timed.

  Here, p is the place connected to the arc a. Moreover, *MS* and *TMS* are untimed and timed colour sets in Σ, respectively.
- $I : P \rightarrow EXPR_\emptyset$ is an initialisation function that assigns an initialisation expression to each place p such that
  - *Type*$[I(p)] = C(p)_{MS}$ if p is untimed;
  - *Type*$[I(p)] = C(p)_{TMS}$ if p is timed.

– $\Pi : T \longrightarrow \{P_{HIGH}, P_{NORMAL}\}$ is the priority function.

In this definition, $EXPR_V$ denotes the set of expressions constructed by using the variables in the set V. Moreover, the initialization expression $I(p)$ for a place p must be a closed expression, i.e., it cannot have any free variables and, as a consequence, $I(p)$ must belong to the set $EXPR_\emptyset$, which is the set of expressions constructed not using variables of the set V, e.g. using just integers.

4 TREE ROUTING Protocol

As commented previously, we compare here three routing protocols by using formal techniques. We estimate that it is more convenient to present them in a chronological order (from oldest to newest) in order to help reader to understand their trade-offs. We start by introducing Tree Routing protocol.

Tree Routing is a protocol for the Network layer in networks with tree topology designed by ZigBee Alliance [Zigbee]. It uses the services of MAC layer which are defined in the standard IEEE 802.15.4. [IEEE 2009] and it is implemented in Castalia Simulator.

4.1 Protocol Description

Tree Routing is a protocol which builds and maintains minimum-cost trees to send information from sensor nodes to base station. It also defines some network parameters which have influence over the tree routing construction. The main parameters are maximum number of devices linked to each router (Cm), maximum number of routers linked to a router (Rm) and maximum depth in the network (Lm). An important restriction is that each node has a 16 bits address so that we can have at most 65.536 nodes associated to it. Furthermore, one can find three types of nodes:

– **Base Station (BS):** In some cases this node is called coordinator or sink. It is the device which collects the data sensed in every node of the network. Depending of the network size, we can use more than one base station. In order to define our model we assume that the system has only one base station.

– **Router (FFD):** They are considered intermediate devices since they gather information from different nodes and forward it upward to another router or to the base station directly. Routers can also sense data from the environment.

– **Leaves (EFD)**. They sense data from the environment and transmit this information upward in the tree.

The three types of nodes are physically different so the network designer has to decide how many routers of each type he uses and where they are situated.

Next, we define the packets used to establish the network:

– **Discovery Packet (DP):** It contains the node identifier and the number of hops from the sender to the base station. It is broadcasted by the base station or a router node and it is received by every node (leaves and routers) in the coverage area of the sender.

– **Join Request Packet (JR):** The packet contains the identifier and the type of the source node and the identifier of target node. It is sent by a node (router or leaf) to another node (router or base station) with which it wants to join.

– **Confirmation Join Request Packet (CJR):** It contains the identifier of the sender and the receiver. It is sent by a router or base station to accept a join request and, it is received by a leaf or other router.

– **Reject Join Request Packet (RJR):** It is similar to CJR packet but it is sent to inform that the join request is rejected.

The tree creation process begins when the base station broadcasts a DP. After receiving the DP, every node waits a predefined period of time to receive new DPs which give more information about neighbours. Router nodes also broadcast a new DP. When a node receives a DP, it saves the information in a table called *neighbour table*. This information contains the number of hops to the base station of the router which has sent the DP. Router nodes calculate the own number of hops to the base station by increasing by 1 the minimum number of hops of its neighbours, which is included in the DP it sends.

After the period of time expires, every node decides which is the best router to join according to the number of hops to the base station and the RSSI (Received Signal Strength Indication). Nodes select the router whose number of hops to the base station is the lowest. If there exists several nodes with the same number of hops, then the selected node will be the one with the highest RSSI because that node is the nearest to the node.

Then, the node sends a JR packet to the selected router in order to join it and it waits for a response. If the request is rejected the selected router is deleted of the neighbour table, selecting a new router to join. If the neighbour table of a node is empty and every join request has been rejected, the node is isolated, that is, the net is not totally connected.

When a router receives a JR packet it checks its configuration parameters (Lm, Cm and Rm) to decide if the join request is accepted or not. Routers accept a join request (and send a CJR packet) if the following conditions hold:

1. The maximum depth of the network (Lm) is not exceeded .
2. The number of nodes (routers and leaves) connected to it is less than or equal to the predefined maximum number of nodes (Cm).
3. The number of routers linked to the router is less than the defined maximum number of routers (Rm).

If one of these conditions does not hold, the request is rejected, and the router sends a RJR packet.

The behaviour of a node in Tree Routing is shown in Fig. 1.

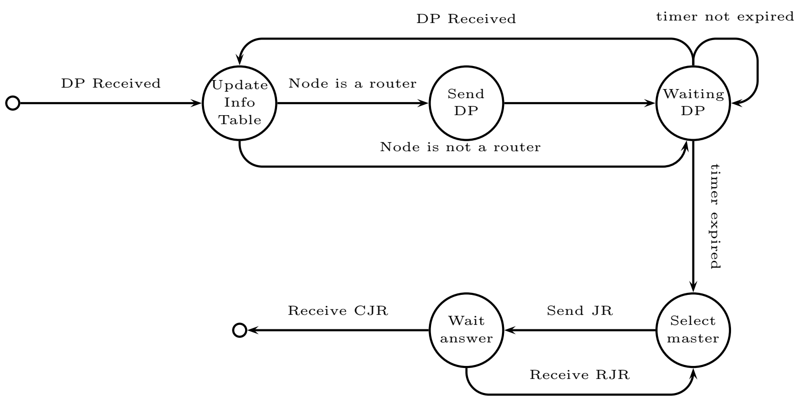


Figure 1: *Node process in Tree Routing protocol*

4.2 PTCPN Model for Tree Routing Protocol

In this subsection we present the PTCPNs of Tree Routing in CPNTools. We use the structure developed in paper [Billington et al. 2009] where an allocation protocol is shown. As a result, a model will expose only one page for representing the WSN, whereas it will state as many *node* pages as the designer wants to check.

Thus, our model is formed by two clearly defined parts. The first one depicts the behaviour of the whole system that is composed by an arbitrary number of nodes and the second one depicts the specific behaviour of each node.

The model for a wireless sensor network with three nodes is shown in Fig. 2. Obviously, this model can be easily generalised to an arbitrary number of nodes. In this model, *Received* place represents the set of packets broadcasted by some node in the system. *Choose Packet* transition selects a packet from this place, marking *Packet* place that represents the packet which has been selected to

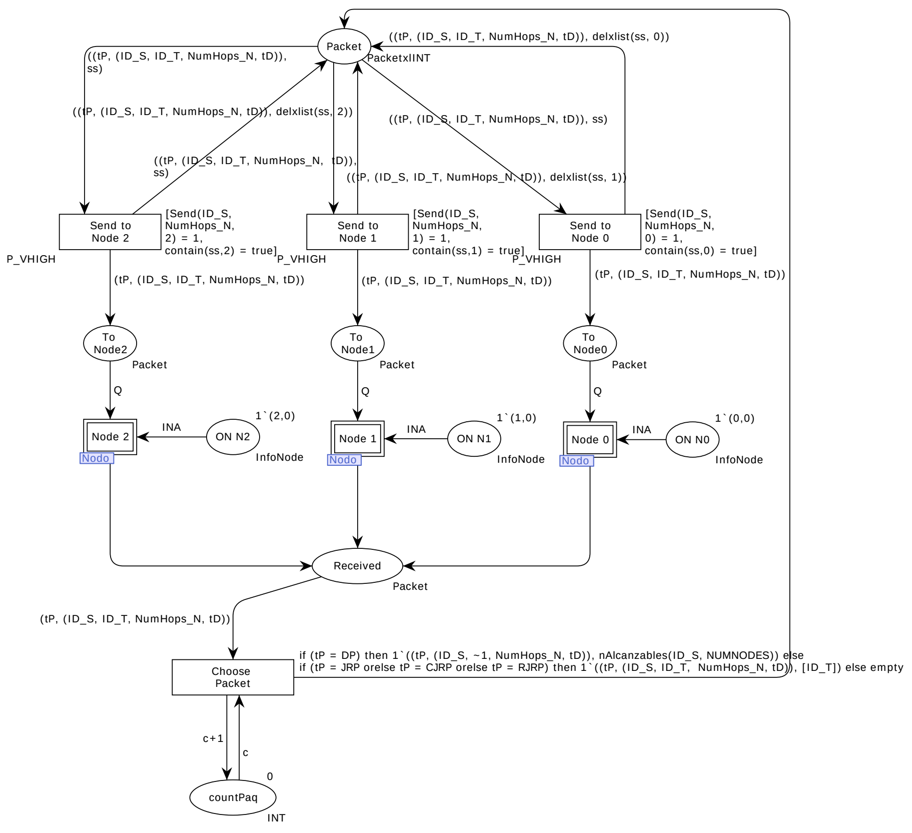

Packet
PacketxlINT
((tP, (ID_S, ID_T, NumHops_N, tD)), delxlist(ss, 0))
((tP, (ID_S, ID_T, NumHops_N, tD)), ss)
((tP, (ID_S, ID_T, NumHops_N, tD)), delxlist(ss, 2))
((tP, (ID_S, ID_T, NumHops_N, tD)), ss)
((tP, (ID_S, ID_T, NumHops_N, tD)), ss)
((tP, (ID_S, ID_T, NumHops_N, tD)), delxlist(ss, 1))
Send to Node 2
[Send(ID_S, NumHops_N, 2) = 1, contain(ss,2) = true]
P_VHIGH
Send to Node 1
[Send(ID_S, NumHops_N, 1) = 1, contain(ss,1) = true]
P_VHIGH
Send to Node 0
[Send(ID_S, NumHops_N, 0) = 1, contain(ss,0) = true]
P_VHIGH
(tP, (ID_S, ID_T, NumHops_N, tD))
(tP, (ID_S, ID_T, NumHops_N, tD))
(tP, (ID_S, ID_T, NumHops_N, tD))
To Node2
Packet
To Node1
Packet
To Node0
Packet
Q
Q
Q
Node 2
Nodo
INA
ON N2
1`(2,0)
InfoNode
Node 1
Nodo
INA
ON N1
1`(1,0)
InfoNode
Node 0
Nodo
INA
ON N0
1`(0,0)
InfoNode
Received
Packet
(tP, (ID_S, ID_T, NumHops_N, tD))
Choose Packet
if (tP = DP) then 1`((tP, (ID_S, ~1, NumHops_N, tD)), nAlcanzables(ID_S, NUMNODES)) else
if (tP = JRP orelse tP = CJRP orelse tP = RJRP) then 1`((tP, (ID_S, ID_T, NumHops_N, tD)), [ID_T]) else empty
c+1
c
0
countPaq
INT

Figure 2: *CPN model for a Tree Routing network with 3 nodes.*

send to one node. When this place is marked, only one of the *"send to node n"* transitions is executed depending on the guards.

Due to the size of the PTCPN model, we have divided it in two parts shown in Figs. 3,4. Notice that there are places with the same name in both figures. Obviously, these places only connect both parts.

We have highlighted the different parts of the algorithm in each figure to explain the behaviour of the algorithm.

In order to simulate the real world, we consider nodes are started at a different time. This situation is modelled in PART1 of Fig. 3 by the time inscription *@+t* and the code in the transition *initializate node*. This transition marks the *On* place, enabling the transition *On Node*, which represents that the node is turned on. The firing of this transition marks the *Node* place with information about the node. After the sensor is turned on, it can receive a DP packet from the base station. This is modelled by the transition *Receive Discovery Message* in PART3 of Fig. 3, which marks *Discovery Packet* place, and enables the transition *Update Neighbour Table*)(see PART2 of Fig. 3). These transitions are enabled when the sensor receives a packet.

PART4 of Fig. 4 models the timer initialization and PART5 represents the process that every node performs to choose the router to join.

When the timer has finished the *Expired Time* place is marked and the node can execute the *Choose Parent* transition to choose its router and then it sends a JR packet. When a router receives a JR packet, it can execute the process modelled in PART6 of Fig. 4, where *Join Devices* transition represents that a CJR packet has been sent and *Reject Devices* models that a RJR packet is sent.

Finally, PART7 models how a node evolves when it receives a CJR or a RJR packet.

5 NORA Protocol

5.1 Protocol Description

NORA is a distributed routing algorithm that assigns roles to the network nodes using both local and neighbourhood information, and creates energy-efficient routes to the base station. Role assignment lies in assigning different tasks to each node or group of nodes in the network in order to globally improve network performance. Role assignment is an efficient manner of optimising specific parameters such as network lifetime, path length or QoS, while data routing is performed. Moreover, this protocol tries to minimize the number of master nodes necessary to connect all nodes in the network. Each router aggregates all packets received from its leaves before forwarding. So that, the less number of routers, the more aggregation and global energy saving.

NORA evaluates node conditions and assigns roles depending on current node and neighbourhood characteristics. The whole process begins at the base station, and ends at the farthest nodes. Intermediate nodes decide between being leaves (nodes that just send sensed data) or masters (which, in addition to the former, also forward data coming from leaves). Each master or leaf selects the best master, inside their radio range, to forward its data to the base station.

By this, every node in the network is able to send data to the base station either directly, or through master nodes. The protocol establishes minimum routes in terms of energy consumption and efficiency, from every node in the network to the base station.

To perform role assignment and route creation, NORA uses three kinds of messages:

– IPM (Information Propagation Message): includes local information such as node ID, number of hops to the base station, remaining battery. . . .

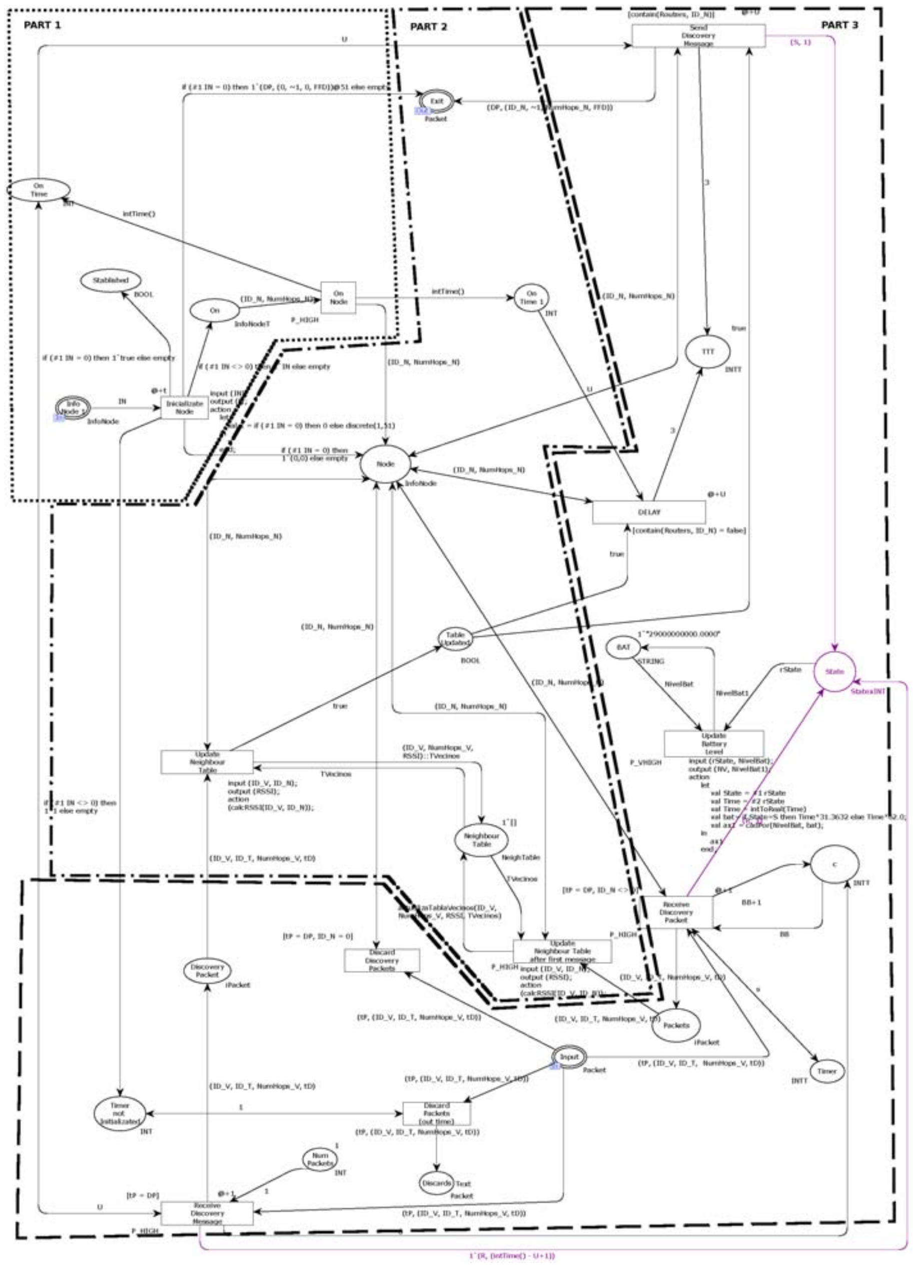


Figure 3: *CPN model for a node in Tree Routing Protocol (part I).*

– RDM (Role Decision Message): includes the same information as IPM, and it is interpreted by nodes as a trigger to initiate the discovery process.

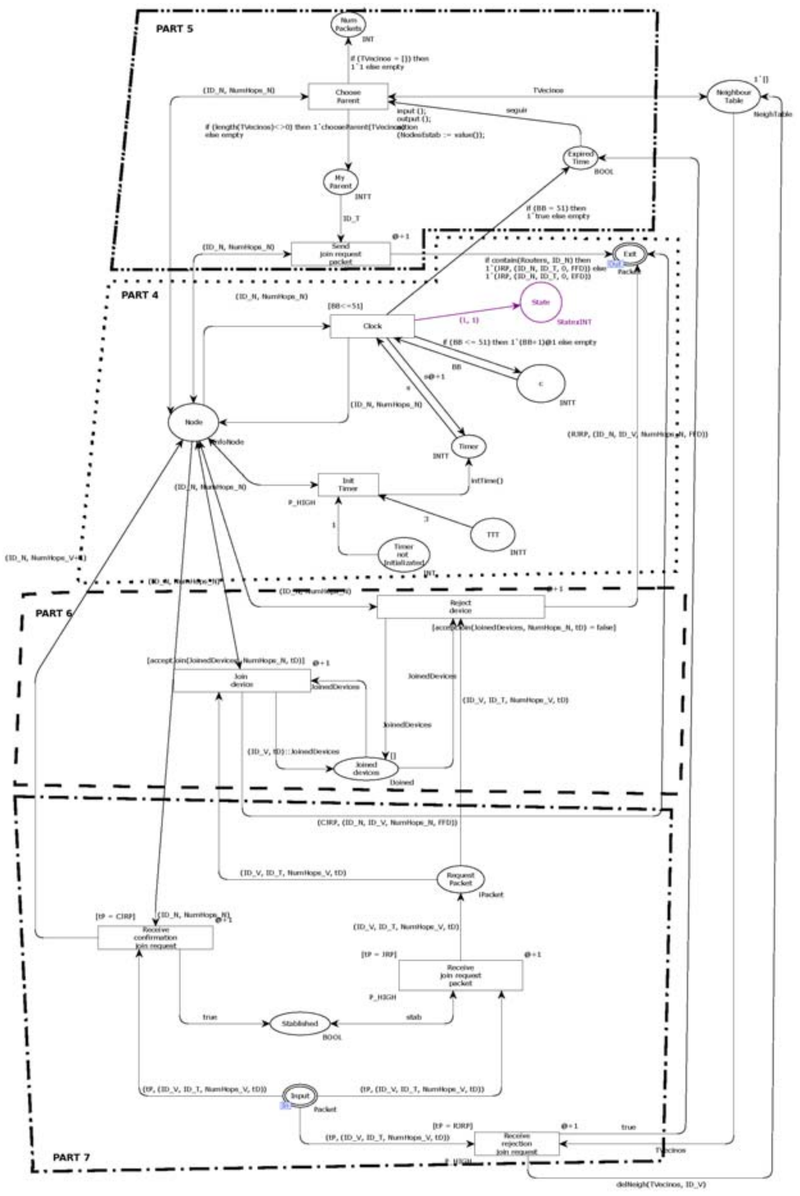


Figure 4: *CPN model for a node in Tree Routing Protocol (part II).*

– MRM (Master Request Message): this kind of message is used by nodes which do not have any master within their radio range (no node can forward their data), and urges a leaf to become master.

The route creation process followed by NORA is outlined in Fig. 5. In NORA, the discovery process begins when the base station sends a RDM. Nodes receiving this message send an IPM and start a timer. During this time interval, nodes wait for information messages from neighbouring nodes. Once the timer expires, nodes perform role decision and master election.

Each node uses the same algorithm as in Tree Routing protocol to obtain its number of hops to the base station, that is, each node increases by 1 the minimum number of hops received in an IPM messages.

One hop nodes will choose the base station as master. If no master is found, an MRM is sent to the best leaf neighbour, i.e. the one characterized by the lowest number of hops, and highest battery level. Once nodes have selected role and master, a RDM message is broadcasted in order to induce the next hop neighbours to start the organization process. This RDM also informs the selected master on the new child. This procedure is propagated hop by hop until the furthest nodes are reached and all network nodes have a route to reach the base station.

In contrast to Tree Routing Protocol, the number of leaves and routers per router is not restricted.

The pseudo-code corresponding to the setup phase is shown in Listing 1.

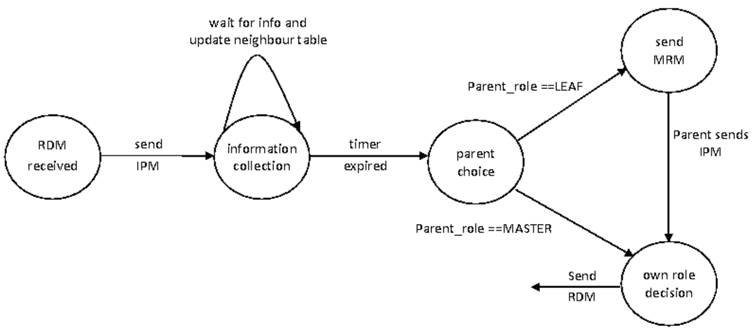


Figure 5: *NORA organization phase transitions*

Below, let us formally define the route creation process. For the sake of readability, we have distinguished between the identifier of the base station, bs, and the identifiers of the nodes, i. Let $ID = \{i \in \mathbb{N}^*\} \cup \{bs\}$ the set of nodes identifiers, we define the function h: $ID \times ID \to \mathbb{N}$ such that $h(i,i) = 0$ and $\forall i,j \in ID, i \neq j, h(i,j) = h(j,i) > 0$ to calculate the number of hops between two nodes. Let $n \in ID$, the set of neighbour nodes, V_n, of the node n with fewer hops number to the base station, bs, is:

$$V_n = \{i \in ID : h(i,n) = 1 \quad \wedge \quad h(i,bs) < h(n,bs)\}$$

Similarly, the set of the same level neighbours (nodes with the same number of hops to the base station as node n), V_{E_n}, is:

$$V_{E_n} = \{i \in ID : h(i,n) = 1 \quad \wedge \quad h(i,bs) = h(n,bs)\}$$

With these previous definitions, master selection process as well as role decision can be defined. Let $R = \{master, leaf\}$ the set of possible roles of a node, l_i the battery level of node i at this moment and $r_i \in R$ the role of the node i, the set M_{V_n} represents the neighbour(s) of the node n with less number of hops to the base station, and whose role is *master*:

$$M_{V_n} = \{i \in V_n : r_i = master \quad \wedge \quad h(i,bs) \leq h(k,bs), \forall k \in V_n\}$$

If the cardinality of this set is just one, such node is selected as master, whereas if the cardinality of this set is greater than one, the algorithm uses the lifetime of the battery to decide which node is the master. Thus, the set $B_{M_{V_n}}$ contains the node(s) of M_{V_n} with the highest level of battery at this moment:

$$B_{M_{V_n}} = \{i \in M_{V_n} : l_i \geq l_k, \forall k \in M_{V_n}\}.$$

Again, if there is only one node in the set, such node is the master, whereas if there are multiple nodes in the set, the node with the lowest identifier is chosen as master. Note that we suppose here the standard total order of the natural numbers. An MRM is sent to the selected node to request it to switch from leaf to master.

On the contrary, if $M_{V_n} = \emptyset$ (no master is in the set of neighbours of the node n), the algorithm follows the same steps, but with the set of leaves:

$$L_{V_n} = \{i \in V_n : r_i = leaf \quad \wedge \quad h(i,bs) \leq h(k,bs), \forall k \in V_n\}$$
$$BL_{M_{V_n}} = \{i \in L_{V_n} : l_i \geq l_k, \forall k \in L_{V_n}\}.$$

Finally, in order to make own role decision, each node compares its battery level with same level (number of hops) neighbours. If its battery level is the highest, then the node sets its role as master. Otherwise, the leaf role is assigned to the node. The function $r(n)$ assigns the corresponding role to the node n:

$$r(n) = \begin{cases} MASTER \text{ iff } \; l_n \geq l_k, \forall k \in V_{E_n} \\ \\ LEAF \qquad Otherwise \end{cases}$$

5.2 PTCPN Model for NORA protocol

The PTCPNs for the implementation of the protocol in CPNTools is presented below. First of all, it is worthwhile to mention that the battery is consumed when a packet is sent or received as well as when the node is waiting for the reception of packets, i.e., each time unit the nodes consume battery. The amount of battery consumed while the node is idle differs from that consumed when the node is sending/receiving packets according to the technical specifications of some commercial nodes such as MicaZ, Telos and so on.

As in the Tree Routing model presented in the previous section, we will present our models following the structure developed in [Billington et al. 2009], showing two clearly defined parts: the *WSN* which depicts the behaviour of the system as a whole composed by an arbitrary number of nodes and the *Node*, that shows the specific behaviour of each node.

In order to reduce the complexity of the model of the whole WSN without decreasing the expressiveness of the model we have made some assumptions about the system. For instance, we have not established a specific representation for the base station since it only participates in the process to start it. Its behaviour is defined by the initial marking (*1'(RDM,(0,0,100,Master))@51*) of the place *Received* in Fig. 6.

Furthermore, in order to simplify the model, the broadcast packet has been modelled with the transition *Packet broadcast* in such a way each node receives its corresponding packets in a kind of "buffer" (Channel Node$_i$), $\forall i \in [1 \ldots n]$, $n = NUMNODES$ (the number of nodes in the system) to temporarily store them before being consumed. The other alternative is to represent the medium as a unique place where all the messages are stored in order to be consumed and this place must be connected with all nodes in the system. This approach is more realistic when the aim is to construct the CPNs for simulating medium access protocols (MAC), whereas we have focused more on the representation of a role decision protocol.

The PTCPN model for a wireless sensor network with three nodes is shown in Fig. 6.

In this model, we have used three hierarchical transitions to enact the nodes. Each node has 2 input places: *Channel Node$_i$ and ON N$_i$*. The channel place is used to store the packets received for a short time. The input arc of this place is labelled with an *if statement*, whose mission is to control that the packet sender is in the coverage area of the node radio (provided for the protocol developers) and its type is *IPM* or *RDM*. When the node has finished its role decision algorithm, the place *FIN N$_i$* is marked in order to visually check that each node has finished, helping us to discover whether a node is stuck. Nevertheless, a place, *Fin* has been added in order to check if all the nodes in the net have finished. Thus, a

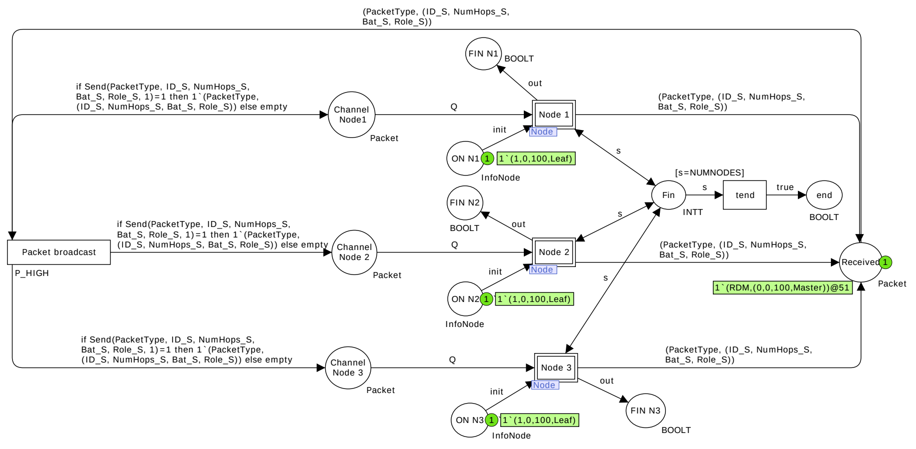

(PacketType, (ID_S, NumHops_S, Bat_S, Role_S))
FIN N1
BOOLT
out
if Send(PacketType, ID_S, NumHops_S, Bat_S, Role_S, 1)=1 then 1`(PacketType, (ID_S, NumHops_S, Bat_S, Role_S)) else empty
Channel Node1
Packet
Q
Node 1
Node
init
(PacketType, (ID_S, NumHops_S, Bat_S, Role_S))
s
ON N1
1
1`(1,0,100,Leaf)
InfoNode
[s=NUMNODES]
Fin
s
tend
true
end
INTT
BOOLT
FIN N2
BOOLT
out
s
if Send(PacketType, ID_S, NumHops_S, Bat_S, Role_S, 1)=1 then 1`(PacketType, (ID_S, NumHops_S, Bat_S, Role_S)) else empty
Packet broadcast
P_HIGH
Channel Node 2
Packet
Q
Node 2
Node
(PacketType, (ID_S, NumHops_S, Bat_S, Role_S))
Received
1
Packet
init
s
ON N2
1
1`(1,0,100,Leaf)
InfoNode
1`(RDM,(0,0,100,Master))@51
if Send(PacketType, ID_S, NumHops_S, Bat_S, Role_S, 1)=1 then 1`(PacketType, (ID_S, NumHops_S, Bat_S, Role_S)) else empty
Channel Node 3
Packet
Q
Node 3
Node
(PacketType, (ID_S, NumHops_S, Bat_S, Role_S))
out
init
FIN N3
BOOLT
ON N3
1
1`(1,0,100,Leaf)
InfoNode

Figure 6: *CPN model for a WSN with 3 nodes.*

parameter *NUMNODES* has been defined, which is initialised with the number of nodes in the system, and when this place is marked with *NUMNODES* tokens, the transition *tend* is fired marking the place *end*. Notice that we have defined the type of this place as *BOOLT*, i.e., a boolean token with a time stamp attached allowing us to extract automatically the time consumed in each simulation.

To model a node we have made some assumptions about the nodes too. As commented above, the amount of battery consumed while the node is waiting differs from that consumed when the node is sending/receiving packets. Moreover, the nodes are started in different moments. The neighbours table is represented as a list that contains the following information at each element: The *ID* of the neighbour, its battery, the number of hops up to the base station and its role. In Listing 1 we show the pseudocode of the algorithm.

Listing 1: NORA and NORIA pseudocode

```
when receive(RDM) then
  update_neighbour_table(RDM)
  send(IPM,BROADCAST)
  temp_neighbours = wait_to_receive(x seconds)
  foreach message in temp_neighbours do
    update_neighbour_table(message)
  end foreach
  temp_master = select_master()
  if get_role(temp_master) = Leaf then
    send(MRM, temp_master)
  end if
  set_master(temp_master)
  RDM_local_role = calculate_role()
  send(RDM_local_role,BROADCAST)
  //start to send environmental data
```

The CPN model of a node has been divided in two parts which are shown in Figs. 7, 8. For the sake of clarity, we have decided to highlight the different parts of the algorithm in each figure. As we said before, nodes with the same name in different figures are the same place in the whole model and they are the conexion between different parts. In Fig. 7 PART1 models the initialization of each node. As NORA developers encouraged, we have opted to start all the nodes at a different time since these devices are not synchronised with respect to their initialization. This situation is depicted by means of the time inscription of transition *Init node*. In CPNTools, a transition with a time inscription *@+discrete(a,b)* means that the output token will have associated a time stamp increases with a random value between a and b. This transition is enabled when the place ONN_i of the system is marked. Nevertheless, the decision process starts when the place *Entry* (Figs. 7, 8) is marked with the token *1'(RDM,(0,0,100,Master))@51* (line 1 of Listing 1) firing the transition *Receive RDM* on PART2 of Fig. 7, which represents the reception of a RDM packet from the base station. As we can see, this token is not enabled until the model time is 51 time units.

Here, we discussed with the protocol designers why the nodes are started before the base station sends the first RDM message. They argued that the nodes need to be ready before the process could occur considering negligible the energy consumed by the nodes up to the process begins. If more than one RDM is received, the place *Rejected* is marked by means of the firing of the transition *Reject RDM*. Next, the neighbours table needs to be updated with the information contained in the packet. The following step is to broadcast the IPMs to the neighbours (line 3 in Listing 1).

The transition *Send IPM* is responsible for marking the place *Exit*, where the tokens are available for being transmitted with the system transition, *Packet broadcast*. Once the messages are sent, a timer (top right corner of PART2 in Fig. 7 and center left area of PART4 in Fig. 8) must be executed during 51 time units in order to gather the information, in form of IPMs, of the neighbours. Going to PART4 of Fig. 8 we have modelled this timer by using the place *c*, which simulates a counter increased either a IPM is received (we suppose that the reception consumes one time unit) or a time unit has elapsed. Thus, for each time unit, the system can elapse a time unit doing nothing or receiving a message.

Here, the transitions *Receive IPM* and *Receive IPM_out* simulate the reception of an IPM, but the main difference between both is that the first one can be fired during that 51 units of waiting for packets and the packets information is used for the master selection, whereas the second one (Receive IPM_out) is used to receive IPMs out of the duration of the timer. Next, according to lines 9 and 10 of Listing 1, the node must consult its neighbour table and select its

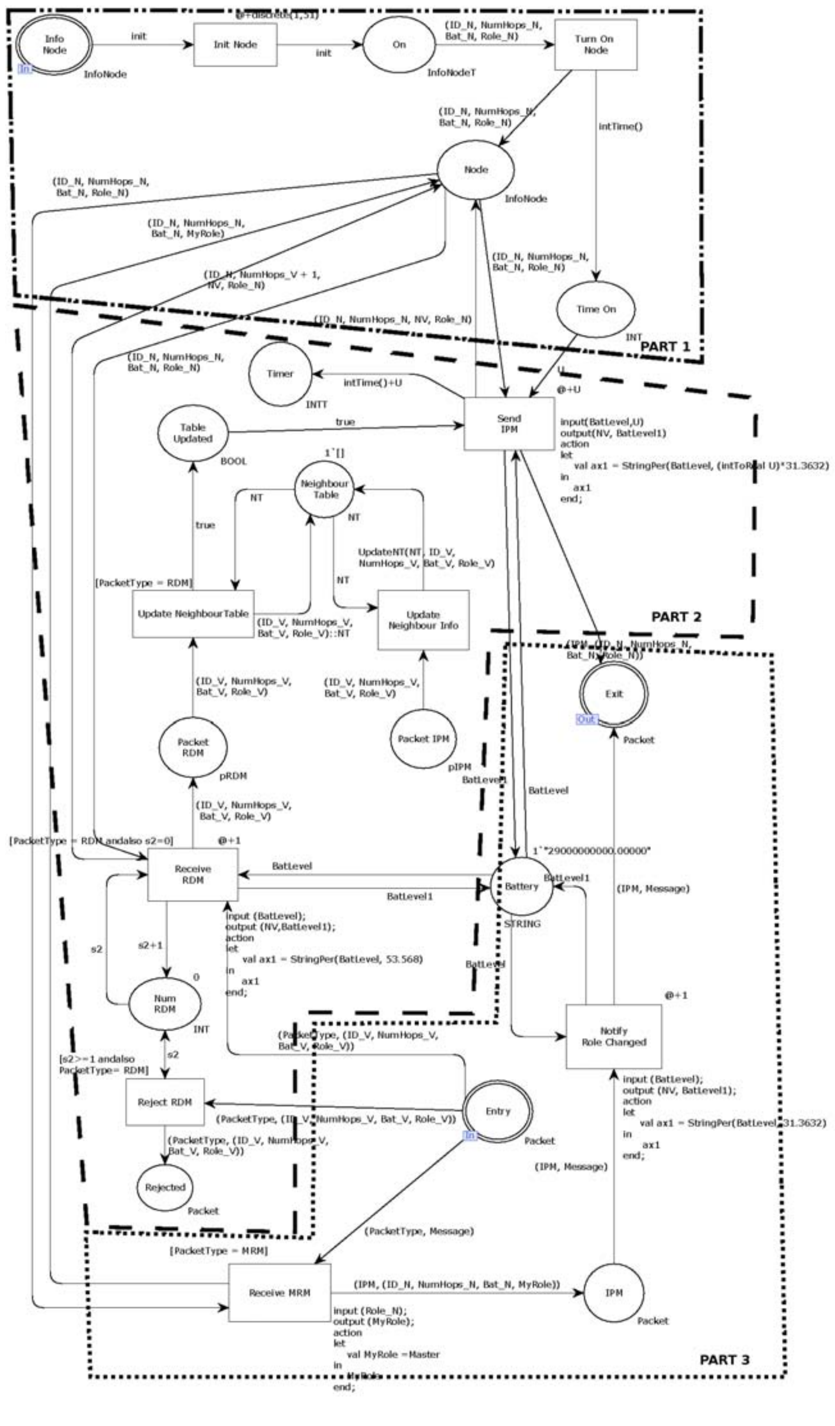
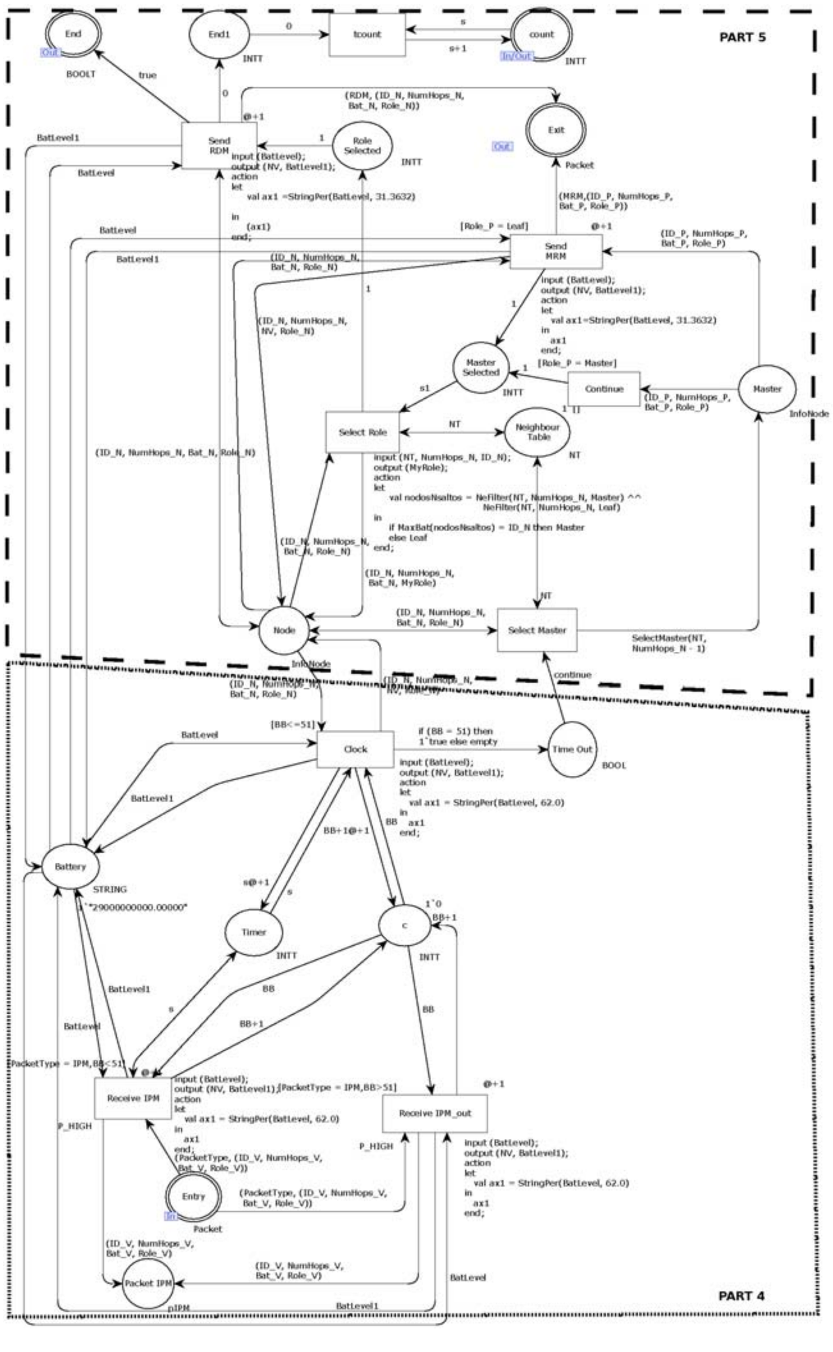


Figure 7: *PTCPN model for a node in NORA Protocol (part I).*



```
input (BatLevel);
output (NV, BatLevel1);
action
let
   val ax1 =StringPer(BatLevel, 31.3632)
in
   (ax1)
end;
```

```
input (BatLevel);
output (NV, BatLevel1);
action
let
   val ax1=StringPer(BatLevel, 31.3632)
in
   ax1
end;
```

```
input (NT, NumHops_N, ID_N);
output (MyRole);
action
let
   val nodosNsaltos = NeFilter(NT, NumHops_N, Master) ^^
                      NeFilter(NT, NumHops_N, Leaf)
in
   if MaxBat(nodosNsaltos) = ID_N then Master
   else Leaf
end;
```

```
input (BatLevel);
output (NV, BatLevel1);
action
let
   val ax1 = StringPer(BatLevel, 62.0)
in
   ax1
end;
```

```
input (BatLevel);
output (NV, BatLevel1);
action
let
   val ax1 = StringPer(BatLevel, 62.0)
in
   ax1
end;
```

```
input (BatLevel);
output (NV, BatLevel1);
action
let
   val ax1 = StringPer(BatLevel, 62.0)
in
   ax1
end;
```

Figure 8: *PTCPN model for a node in NORA Protocol (part II).*

master and its role. On the one hand, the master selection is done by the transition *Select Master* of PART5 (lower right corner of Fig. 8) and by the function *select_master*. This function encodes the algorithm for master selection. If one master is found in the neighbours table, the node will continue selecting its role with the code accompanying the transition *Select Role* (lower right corner of the transition). In CPNTools, each transition may have an attached code segment which contains ML code, executed when the transition is fired. This code implements the algorithm for role decision. Nevertheless, if no master is found, the node must send a MRM (master request message) to its best neighbour leaf continuing with the role decision process as if a master has been found. Looking at PART3 of Fig. 7, when the MRM is received (transition Receive MRM), the node sends an IPM in order to notify to its neighbours that its role has changed. Finally, once the master and role decision processes are finished, the node sends a RDM (Send RDM transition) to start the decision process in the next level and increments the counter of finished nodes. Let us note that the transition *tcount* and the place *count* are depicted with accountability purposes since they store transiently the number of nodes that have finished.

As commented in the introduction, NORA (and NORIA) include some mechanisms to manage situations related to node failures, new nodes coming to the network, and low-resourced routers. Notice that these mechanisms are generally used in the maintenance phase, although they could be required in the setup phase. Next, we present how this task is done in NORA and how this could be done in our model. NORIA and NORA use the same mechanisms to deal with these situations:

- **Low-resourced master**: in order to model master failures and network division, when the battery level of a master node drops below a threshold, it requests a device type (its role will be leaf from now on) change and its child nodes will look for another master. Thus, child nodes are able to select another parent, and network connectivity is preserved. This can be easily implemented in our PTCPN model by controlling the battery level of master nodes (obtained from the place *Battery*), and including an extra message that it is send when this threshold is reached. When receiving this message, leaves can select the best master in their neighbours table using the same method as in the route creation.
- **Master failure:** master nodes send acknowledgement messages (ACKs) when receiving data from other nodes. If a node does not receive the ACK message from its parent during two consecutive times, it will proceed as if it had received a device type change request from its parent node. We can include these ACKs as tokens in the place *Entry* and include the necessary conditions to consume these tokens when required (in a similar way as for other messages, see part 2 and 3 for example).

– **New nodes joining the network:** when the network is already in operation, a new node willing to join the network should first monitor the channel. If any message from the base station or a master is detected, the node selects it as parent, updating its neighbour table. If no master or base station message is received, the new node will send a MRM request to the best device heard in a predefined time, selecting it as parent. By default, the role of the new node will be leaf. The nodes can act in our model as when no master is found in the neighbour table in the setup phase. Let us comment we must include a new transition, whose input place is *Entry* as in the reception of a RDM or a MRM message. Then, we can update the neighbour table and select the master of this newly added node as it is done when a RDM is received.

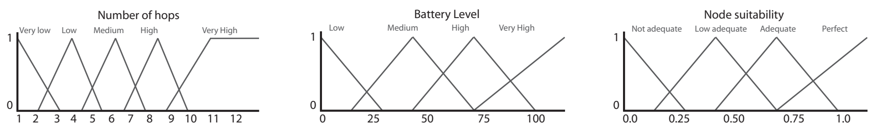
6 NORIA Protocol

NORIA is a distributed routing algorithm for WSNs that improves NORA protocol. The main difference between both protocols is that NORIA uses fuzzy logic to assign the roles in the network.

Fuzzy logic is a decision system approach which works similarly to human control logic [Mendel 1995]. It provides a simple method for reaching a conclusion from imprecise, vague, or ambiguous input information. The execution of a fuzzy-logic system requires less computational power than conventional mathematical computational methods such as addition, subtraction, multiplication and division. Furthermore, only a few data samples are required in order to extract the final accurate result. Besides, fuzzy logic is a handy technique since it uses human language to describe inputs and outputs.

The route creation followed by NORIA is similar to NORA (see Fig. 5 and Listing 1). In this case function *SelectMaster*, in the output arc of transition *Select Master* (PART5 of Fig. 8), selects the lower level (lower number of hops to the base station) neighbour with best fuzzy logic evaluation value. In addition the attached code to the transition *Select Role* decides the own role by comparing the own fuzzy logic evaluation value.

Master selection and role assignment are based on the results of the evaluation of a fuzzy rules set. To perform the role assignment and master selection process, nodes will compare the evaluation output for each neighbour node. The input variables to be considered in the experiments are: number of hops to reach the base station and remaining node energy. These parameters are just a subset within the full set of parameters which can be included in the decision process (delivery probability, delay or signal strength among others). The output variable represents the suitability of the node for being a master or for being selected as master node. Fig. 9 shows fuzzy sets for input and output parameters.



(a) Number of hops. (b) Battery level. (c) Output variable

Figure 9: *Input and output Fuzzy sets.*

The fuzzy sets used in this paper are an example of the multiple possibilities and have been designed after several checks for the application and topologies used, in order to have a generic proposal which works in a wide range of WSN applications. It is important to note that fuzzy sets (input and output) can be customised depending on the application, requirements and circumstances of each particular WSN. For example, in a network that needs real-time data collection, the usage of the end-to-end delay as a decision parameter would be useful. For that example, the fuzzy rule base includes rules such as the following: "IF Number of Hops is Low AND Battery Level is High THEN Node Suitability is Adequate". Here, since we have 4 fuzzy sets for battery level input and 5 for number of hops input, in total we have 20 rules.

7 Analysis and Performance Evaluation

So far, we have introduced three routing protocols: Tree Routing, NORA and NORIA, as well as the corresponding PTCPNs.

Before we begin their study, it is required to present the scenarios in which the results have been collected. The radius of the coverage area for each node is 50m. The designed scenario considers circular network areas with radius from R (50m) to 3R (150m), where nodes may be deployed randomly in the case that one wants to obtain performance results for an irregular topology or, on the contrary, the nodes may be placed maintaining a constant node density. The coordinator is located at the centre and the number of nodes in the network depends on the radius. Fig. 10(a) shows the distribution of sensors in 50m radius-area where we have placed 2 routers and 18 sensors, Fig. 10(b) shows the distribution of 81 sensors in a 100m radius-area, where 31 are routers and 49 are sensors, and, finally, Fig. 10(c) shows the distribution of 177 sensors in a 150m radius-area where we deploy 40 routers and 136 sensors.

We exploit the two possibilities offered by CPNTools (simulation and formal verification) to ensure the correctness of the protocol under study.

Regarding to formal verification, let us note that it is based on an exhaustive exploration of the whole system state space, that is, CPNTools, in our case, explores all possible paths in the model. This approach suffers frequently the

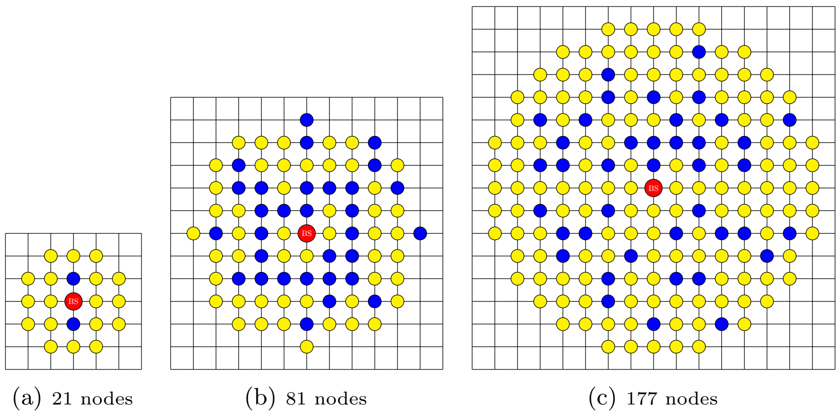


Figure 10: Topology of regular network

well-known problem called state-space explosion. In this case, we have experimented this problem, and, therefore, we have conducted the analysis of the state space with networks with nodes from 3, 4 (full) and 5 (partial). In Tab. 1, we show the results for the protocol NORA.

Properties / No. of Nodes	3	4	5
State Space Nodes	2863	39687	225444
State Space Arcs	4982	128428	671466
Time (hh:mm:ss)	00:00:3	00:01:38	72:00:00
Dead Markings	6	24	-

Table 1: *State space analysis results*

Now, we are able to assert in this case that the system works correctly, since it has been checked that all the dead markings in full state space includes the marking of the system place *end*. To this end, we have implemented the function *fun DesiredTerminal n =((Mark.Sistema'end 1 n) == 1'true)*, defined in [Jensen and Kristensen 2009], that returns the state space nodes where the place *end* is marked with the boolean *true*: in addition, we have evaluated if the other function *PredAllNodes DesiredTerminal=ListDeadMarkings()* returns *true* in order to check if all the dead markings nodes of the state space hold that predicate.

On the other hand, the number of Strongly Connected Components (SCCs) is the same as the state space according to the statistical report generated by

CPNtool which demonstrates the absence of cyclic behaviours as expected.

As we commented previously, our model suffers the state space explosion problem so that we must exploit the other ability provided by CPNTools. Thus, we have conducted several experiments in order to obtain interesting results about the performance of these protocols. Before we begin, it is required to introduce briefly the most important technical details for a better understanding. The machine features in which we have performed the experiments are the following: AMD Phenom II 3.3 GHz, 16 GB RAM.

Furthermore, the evaluation of these proposals is based on the following performance metrics:

1. Number of nodes associated in the network: this parameter is crucial to system operation since if the network is not totally connected it is impossible to collect data from leaves to the base station, that is, there are some parts of field that are not covered.

2. Battery consumption: obviously, it is one of the most important parameter since it allows us to calculate approximately the lifetime of the net. Moreover, battery replacement may be laborious and sometimes impossible, so the energy saving has to be considered when evaluating a protocol for wireless sensor networks.

3. Network set-up time: the time that the network takes to be ready to start sending application data, that is, the time that all network nodes take to decide device type and parent node.

4. Average number of packets sent during set-up process: it is the average number of packets sent by all nodes in the network during set-up stage. Since the number of packets sent is directly related to energy consumption, this metric is interpreted as a measure of the energy consumption. In addition to this, this could help to reduce the number of retransmissions in the MAC layer.

5. Number of master nodes after set-up phase: it is the number of master nodes necessary to connect all nodes in the network. Data aggregation allows the reduction of data forwarding in the network, and, therefore, the less number of routers in the network, the more aggregation and less amount of data to be transmitted in the network, achieving global energy saving.

For each scenario and particular combination of parameters, we run 100 simulations in order to obtain reliable results. We must point out that in order to make the most out of this study, the comparison has been divided into two parts, comparing pairs of protocols which allowed us to observe the improvements obtained with each protocol.

Given that NORA protocol emerged as a possible improvement to the TR protocol, it seems natural to present a comparison between them in order to check if any improvement has been achieved. Thus, with the data collected from the 100 simulations for each scenario listed in Tab. 2, and having regard to the performance metrics discussed above we can say that the main difference/improvement of NORA protocol with respect to its predecessor is the fact that NORA achieves full connectivity of all network nodes in all cases.

Algorithm	Network Size	Topology	Associated	Battery (J)	Setup Time (ms)	Packets	Battery (Avg)
Tree Routing	21	Regular	21	0.14577582	162.03	59	0.00694171
Nora	21	Regular	21	0.13688740	153	44	0.00684437
Tree Routing	21	Iregular	21	0.1350794	152.92	41	0.00643235
Nora	21	Irregular	21	0.120966	152	44	0.00604831
Tree Routing	81	Regular	33	0.5708123	202,01	166	0.00704706
Nora	81	Regular	81	1.130445	267	206	0.01413056
Tree Routing	81	Irregular	35	0.5693042	202.22	151.53	0.00702844
Nora	81	Irregular	81	1.1965692	282	209	0.01495712
Tree Routing	177	Regular	35	1.38274678	240.42	265.87	0.007812124
Nora	177	Regular	177	3.4904808	361	475	0.01972023
Tree Routing	177	Irregular	38	3.3702193	241.53	268.02	0.01904078
Nora	177	Irregular	177	3.2090162	367	453.723	0.0181300339

Table 2: *Perfomance Results for Tree Routing and NORA protocols*

This was the main motivation for the development of NORA protocol. For small networks (21 nodes in the table), the TR protocol manages to connect all nodes, when the network grows up to sizes that can be found in real systems TR protocol does not reach a 50% connectivity in any case. In networks with 81 nodes, TR protocol achieved only 40.74% and 43.20% of connectivity for regular and irregular topologies respectively and in networks with 177 nodes, this connectivity falls dramatically to 19.77% and 21.46%. This fact is obviously worrying because it demonstrates that there are parts of the area covered by the sensors which are not getting data. This issue has a direct influence in the rest of the parameters evaluated. Focusing on battery consumption we can see that in the case of networks with 21 nodes, where both protocols achieve full connectivity, the total battery consumption and the average one is lower for NORA protocol (for both types of topology).

This happens for the rest of parameters to evaluate when the TR protocol achieves full connectivity (NORA protocol always gets it), values for both the setup time and the number of packets sent is greater than those obtained in the protocol NORA . Showing clearly additional improvement to the main: get full connectivity. For example, consider the case of a network with 81 nodes in regular topology. The protocol TR takes 202.01 ms to establish a network connecting 33 of the 81 nodes. The protocol NORA takes more time (495.84 ms) but manages to connect the 81 nodes. What does this mean? The system will detect the failure of nodes in the network established which will require further executions of this protocol in order to reconfigure the network. This obviously will take more packet traffic, spending more time and battery.

The last of the performance metrics established before, number of master nodes after set-up phase, can not be used to compare these two protocols as NORA protocol is based on roles, and the protocol itself will decide the role of each node according to needs and therefore the number of both master and leaves. But TR protocol is not based on roles, there are master nodes and leaf nodes with distinct features and functions and the data obtained will depend on the exact spatial location of each one. Therefore, to allow comparison between the two protocols for the rest of the parameters, an initial configuration for each network size studied has been established which have been presented previously and that can be seen in Figs. 10(a), 10(b) and 10(c).

Next, we compare NORA and NORIA taking into account the same scenarios and analysing the same measures. For each scenario and particular combination of parameters, we run again 100 simulations. The results of our experiments are depicted in Tab. 3. To begin with, in column *Associated*, it can be observed that both protocols achieve fully connectivity, that is, all the nodes deployed in the grid are able to associate with other nodes in order to collect sensed information and forward it to the base station. This is one of the key differences with Tree Routing. Notice that the inclusion of fuzzy logic does not degrade the connectivity in the net. We commented that energy is a key metric in the design of a protocol for wireless sensor networks owing to nodes are highly energy constrained. Since the wireless interface in the part that consumes most energy in the nodes, the number of sent packets is directly related to the energy consumption. The number of sent packets and the battery consumption when adding fuzzy logic to the decision making system is mainly equal to using NORA, and, therefore, we can assert that the addition of a fuzzy-logic based decision system does not imply a higher energy consumption. Nevertheless, it is worth to mention here that there is parameter that influences, to some extent, both measures. As it can be observed in Tab. 3, the energy consumption for the irregular topology is, in most of the cases, greater than the one for the regular topology. This is owing to the fact that when nodes are deployed randomly in the field, the probability of not finding a master among your neighbours is higher than in the regular topology and, as a consequence, nodes require to send a MRM to their best leaf in order to encourage it to become a master node, leading to a higher energy consumption. Finally, regarding to the number of masters after set-up phase, column *Master* shows that the protocol NORIA is slightly worse due to the number of masters are higher. Interestingly, NORIA obtains always worst results when the topology under study is irregular, whereas NORA yields worst

Algorithm	Network Size	Topology	Associated	Battery	Setup Time	Packets	Masters	Leaves
Nora	21	Regular	21	0.1243324	152.94	42.8	3.05	17.95
Noria	21	Regular	21	0.127459192	153.342	42.597	2.597	18.403
Nora	21	Irregular	21	0.100159097	154.05	43.061	4.061	16.939
Noria	21	Irregular	21	0.127592018	153.247	44.058	4.058	16.952
Nora	81	Regular	81	0.727286356	282.35	211.645	32.112	48.889
Noria	81	Regular	81	1.177434	282.154	211.719	32.102	48.898
Nora	81	Irregular	81	0.728669645	282.278	199.123	25.388	55.612
Noria	81	Irregular	81	1.17584715	282.497	199.276	25.51	55.49
Nora	177	Regular	177	3.3752254	350.754	470.39	64.715	111.285
Noria	177	Regular	177	3.2874375	347.765	455.24	62.725	115.275
Nora	177	Irregular	177	3.2100174	349.987	452.915	57.525	118.475
Noria	177	Irregular	177	3.41732741	357.676	466.5	60.735	116.265

Table 3: *Performance Results for NORA and NORIA protocols*

results when the topology is regular. Therefore, we can conclude that NORA is a slightly better choice when the topology is irregular, and NORIA otherwise.

After evaluating the performance of both approaches, the main conclusion one can observe is that the inclusion of fuzzy logic in the decision making process neither degrade nor improve the performance of the system.

8 Conclusions and Future work

In this work, we have formally studied three routing protocols. To this end, we have used a rigorous and neat formalism, prioritised-timed coloured Petri nets, and a mature tool such as CPNTools to analyse them. As usual, we have conducted some experiments in order to compare these protocols, showing that NORA and NORIA obtain better results when the net is totally connected. This is an important result for protocol designers since many real applications need full connectivity, e.g., critical or emergency systems. Moreover, we have analysed by means of formal verification if NORA and NORIA have any undesirable property such as a deadlock. The results obtained showed that both proposals are deadlock-free when the net is not very large. Unfortunately, we could not obtain concluding results for bigger nets due to the state space explosion suffered by both models. As future work, we could study other properties such as liveness or soundness. We could also extend both algorithms to deal with probabilities to model more realistic scenarios in which the packets could be lost. Finally, we are currently working on the development of a tool to automate the construction of bigger models.

Acknowledgements
This work has been partially supported by CICYT project TIN2012-36812-C02-02

References

[CPNtool 2013] CPNTools web site. http://cpntools.org, jan 2013.

[Billington et al. 2009] J. Billington and C. Yuan. On modelling and analysing the dynamic manet on-demand (dymo) routing protocol. In *Transactions on Petri Nets and Other Models of Concurrency III*, volume 5800 of *Lecture Notes in Computer Science*, pages 98–126. Springer Berlin / Heidelberg, 2009.

[Espensen et al. 2008] K. L. Espensen, M. K. Kjeldsen, and L. M. Kristensen. Modelling and initial validation of the dymo routing protocol for mobile ad-hoc networks. In *Proceedings of the 29th International Conference on Applications and Theory of Petri Nets*, ICATPN '08, pages 152–170. Springer-Verlag, 2008.

[IEEE 2009] IEEE Standard for Part 15.4. Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (WPANS). http://standards.ieee.org/getieee802/download/802.15.4c-2009.pdf.

[Jensen and Kristensen 2009] K. Jensen and L. M. Kristensen. *Coloured Petri Nets - Modelling and Validation of Concurrent Systems.* Springer, 2009.

[Mendel 1995] J. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83:345–377, 1995.

[Ortiz 2011] A. M. Ortiz. Intelligent routing techniques for wireless sensor networks. Doctoral Thesis, 2011.

[Ortiz et al. 2011] A. M. Ortiz, F. Royo, T. Olivares, L. Orozco, J. C. Castillo, and A. Fernández. Protocol integration for intelligent monitoring applications in wireless sensor networks. In *Proceedings of the 4th international conference on Interplay between natural and artificial computation - Volume Part I*, IWINAC'11, pages 511–520. Springer-Verlag, 2011.

[Park et al. 2000] S. Park, A. Savvides, and M. B. Srivastava. Sensorsim: a simulation framework for sensor networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWIM '00, pages 104–111. ACM, 2000.

[Pham et al. 2007] H. N. Pham, D. Pediaditakis, and A. Boulis. From simulation to real deployments in wsn and back. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6, 2007.

[Ruiz et al. 2012] M. C. Ruiz, J. A. Mateo, H. Macia, J. J. Pardo, and T. Olivares. Formal modelling and performance evaluation of a novel role-based routing algorithm for wireless sensor networks. In *Proceedings of the 18th annual International Conference on Advanced Computing and Communications*, ADCOM 2012, pages 98–104. IEEE, 2012.

[Yue et al. 2010] H. Yue, H. Bohnenkamp, and J. P. Katoen. Analyzing energy consumption in a gossiping mac protocol. In *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, volume 5987 of *Lecture Notes in Computer Science*, pages 107–119. Springer Berlin Heidelberg.

[Yue and Katoen 2010b] H. Yue and J. P. Katoen. Leader election in anonymous radio networks: Model checking energy consumption. In *Analytical and Stochastic Modeling Techniques and Applications*, Lecture Notes in Computer Science, pages 247–261. Springer Berlin Heidelberg.

[Zigbee] Zigbee Alliance. Zigbee Specification. http://www.zigbee.org.