Real deployment of Consensus Algorithm on Self-organized Wireless Sensor Networks

ELBHIRI Brahim
Laboratoire de Recherche en Informatique et Telecommunication
FSR Rabat, Maroc
Email: elbhirij@yahoo.fr

Alba Pagès-Zamora
Signal Processing and communications group
UPC, Barcelona, Spain
Email: alba.pages@upc.edu

Driss Aboutajdine
Laboratoire de Recherche en Informatique et Telecommunication
FSR Rabat, Maroc
Email: aboutaj@fsr.ac.ma

***Abstract*—[1] Reaching consensus on a self-organized wireless sensor networks through totally decentralized algorithms is a topic that has attracted considerable attention. The average consensus method is the most popular algorithm used in this kind of applications. The main advantage of these approaches is that the network does not involve a fusion center to organize nodes. Using a realistic environment to check the behavior of this scheme is the major objective of this work. Moreover, this paper contributes to answer and confirm some results which are approved by theoretical works.**

I. Introduction

A wireless sensor network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations [1]. Consensus problems also called agreement problems, have a long history in the field of computer science, particularly in automata theory and distributed computation [2]. They are typically found in decentralized systems, where sensors interact with each other to reach an agreement regarding a certain value that depends on the state of the sensors. One of the most critical aspects of these consensus algorithms lies on the fact that they are iterative algorithms where, at each step, the network nodes exchange data among each other to achieve agreement [3]. There is a substantial amount of theoretical works which explain and clarify the importance of this consensus algorithm than a centralized system [4]–[7]. But, there are not lot of works that check and improve these results on a realistic environment. Castalia is a Wireless Sensor Network (WSN) simulator based on the OMNet++ platform that can be used to test this distributed algorithm, particularly consensus algorithms, using a realistic wireless channel and radio model, with a realistic node behavior especially in relation to the radio access and the wireless channel. Moreover, it take into account some of the effects of the drifting clocks and introduce the collision problem based on an additive interferences model [8]. These interferences are dynamically calculate from different transmitting nodes and thus calculating dynamically the Signal to Noise Ratio (SNR) and the resulting packet reception probabilities. A floor noise is introduced in third elements of nodes, in the sensing device, in the radio and in the wireless channel model. On [9], L. Pescosolido and all take into account the inevitable noise present in the interaction among the sensors through realistic propagation channels, but they not propose a collision avoidance technique. In our paper we have use a simple random MAC with a simple strategy to control usefully this collision problem. An outline of this paper is as follows. In section II a graph definitions and the basics of graph theory are given, In section III, a brief overview of the consensus algorithm that has been implemented is given. In section IV, we present a small description of Castalia software. Additionally, in section VII, a brief description of the way in which nodes are programmed in our simulations is provided. Finally, the simulations and conclusions are presented.

[1]This work has been partially funded by the European Commission $(033914-WINSOC)$, the Hassan II academy of sciences and technology and the Cooperation project $(A/8885/07)$ between LRIT of the Mohammed V-Agdal University $(Rabat, Morocco)$ and TSC department from UPC $(Barcelona, Spain)$

II. Graph definition

The information flow in a wireless sensor network can be described by means of a graph, where the nodes represent the sensors and the edges correspond to the communication links among the sensors. A graph G is defined as G=$\{$V,E$\}$, where V is a set of nodes (or vertices) indexed with $i = \{1, ..., N\}$ and E is the set of undirected edges e_{ij} between nodes i and j. We assume a graph without loops or multiple edges. The set of neighbors of node i are given by $N_i = \{1, ..., N\}$. The adjacency matrix of the graph G, denoted $\mathbf{A} \in \Re^{N\times N}$, has entries equal to

$$\mathbf{A} = [a_{ij}] = \begin{cases} 1 & \text{if} \quad e_{ij} \in \text{E} \quad \forall i, j = \{1, ..., N\} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The out-degree of a node corresponds to the total number of incoming edges, $\sum_{j=1}^{N_i} a_{ij}$. The degree matrix $\mathbf{D} \in \Re^{N\times N}$ of G is a diagonal matrix with diagonal entries equal to the out-degree of each node. The $N \times N$ Laplacian matrix of the graph is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2}$$

III. The consensus algorithm

The consensus algorithm is the well-known iterative algorithm presented by Olfati-Saber and Murray in [6]. In this

algorithm each node of the network is programmed to run a discrete dynamical system whose state evolves according to the following difference equation:

$$x_i(k) = x_i(k-1) + \epsilon \sum_{j \in N} a_{ij}(x_j(k-1) - x_i(k-1)) \quad (3)$$

In 3, $x_i(k)$ denotes the state of node i at time k, ϵ is a positive scalar selected in the interval (0,1/N-1] to satisfy convergence conditions and N_i is the set of neighbors of node i. Let $\mathbf{x}(k)$ denote the vector of all states at time $k > 0$ and assuming a time-invariant network topology, the evolution of $\mathbf{x}(k)$ can be written in matrix form as follows:

$$\mathbf{x}(k) = \mathbf{W}\mathbf{x}(k-1) \quad (4)$$

where $k > 0$ and $\mathbf{x}(0) = [x_1(0)...x_N(0)]^T$ and the weight matrix $\mathbf{W} \in \Re^{N \times N}$ is modeled as

$$\mathbf{W} = \mathbf{I} - \epsilon \mathbf{L} \quad (5)$$

Where is $\mathbf{L}$ the Laplacian of the graph and $\mathbf{I}$ is the identity matrix. If the network is connected, the state of each node in the network will converge to the average of the initial values,

$$\lim_{k \to \infty} x_i(k) = \frac{1}{N}\mathbf{1}^T\mathbf{x}(0) = \frac{1}{N}\sum_{j=1}^{i=N} x_i(k) \quad (6)$$

and we say that the network reaches the average consensus. We propose an implementation of the iterative algorithm in 3 in which the nodes exchange packets with its own state encoded and certainly some extra useful information. Notice that an exact implementation of equation 3 would require that all nodes update their state in a synchronous and coordinated way. However, we are looking for a wireless sensor network composed of simple nodes with limited processing capabilities and few networking features. This is to reduce the extra signalling among the nodes and to enhance the network scalability. Because we implemented an iterative approach, we propose an extremely simple Medium Access Control scheme in which the nodes keep listening the channel so that each time one node receives a packet from a neighbor it stores the state of that neighbor in a table. After a certain time, the node updates its state using the states of the neighbors stored in its table, and broadcasts its own new state. This procedure of listening and transmitting is iterated periodically in time. But, so far, we are able to ensure that the implemented consensus algorithm will not exactly match equation 3 because of two reasons. First, the topology of the network will vary with time because the communication links between nodes may fade or the packets sent by two neighbors may collide. Therefore, we cannot guarantee that the same set of neighbors are available each time a node updates its state. Second, a classical assumption in 3 is that W is a symmetric matrix. In a practical implementation this would require an acknowledgement protocol to guarantee that, at each iteration, a given node i updates its state using only the states of those neighbors that have also received the previous state of the i^{th} node. The weight matrix W will be a symmetric matrix if for each pair of nodes (i,j) the node i uses the state of node j and respectively node j uses the state of node i to update their states. Summing up, the consensus algorithm implemented in Castalia will better fit the model:

$$\mathbf{x}(k) = \mathbf{W}(k-1)\mathbf{x}(k-1) \qquad k > 0 \quad (7)$$

Where $\mathbf{W}(k)$ is the weight matrix at time k which varies from one iteration to another.

This model assumes a time-varying network with random topology. A time-varying network can be characterized by a dynamic graph defined at time k as G(k)= {V, E(k)}, where E(k) is the instantaneous set of edges e_{ij} for all $i, j = \{1, ..., N\}$. Moreover, when the topology is random, the existence of a link between any pair of nodes is probabilistic and we say that $e_{ij} \in$E(k) with probability $0 \leq p_{ij} \leq 1$. Let us define the connection probability matrix $\mathbf{P} \in \Re^{N \times N}$ with entries $[\mathbf{P}]_{ij} = p_{ij}$ and $[\mathbf{P}]_{ii} = 0$. $\mathbf{A}(k)$ is the instantaneous adjacency matrix of a time-varying network with a matrix connection probability $\mathbf{P}$. The $\mathbf{A}(k)$ is a equal to:

$$\mathbf{A}(k) = [a_{ij}] = \begin{cases} 1 & \text{with probability } p_{ij} \\ 0 & \text{with probability } 1 - p_{ij} \end{cases} \quad (8)$$

The Laplacian matrix of the underlying graph is also random and given by:

$$\mathbf{L}(k) = \mathbf{D}(k) - \mathbf{A}(k). \quad (9)$$

Thus, the time-varying weight matrix is modeled as:

$$\mathbf{W}(k) = \mathbf{I} - \epsilon \mathbf{L}(k) \qquad k > 0 \quad (10)$$

The main results obtained in 3 are derived assuming that the initial set of measurements are random with known mean, $E[\mathbf{x}(0)] = x_0\mathbf{1}$, and variance σ_0^2 [10]. In this case we can ensure that the expected value of the state vector will converge to the mean average vector, that is

$$\lim_{k \to \infty} E[\mathbf{x}(k)] = \mathbf{x}(0)\mathbf{1} \quad (11)$$

IV. CASTALIA

Castalia is a Wireless Sensor Network (WSN) simulator based on the OMNet++ platform that can be used by researchers and developers who want to test their distributed algorithms and/or protocols in a realistic wireless channel and radio model, with a realistic node behaviour especially relating to access of the radio. Castalia can also be used to evaluate different platform characteristics for specific applications, since it is highly tunable, and can simulate a wide range of platforms [8]. The main features of Castalia are:

- Advanced channel/radio model based on empirically measured data.
- Detailed state transition for the radio, allowing multiple transmission power levels.
- A highly flexible physical process model. Sensing device noise, bias, and power consumption.
- Node clock drift, CPU power consumption.
- Resource monitoring that goes beyond energy consumption (such as memory and CPU time).
- A highly configurable MAC layer protocol

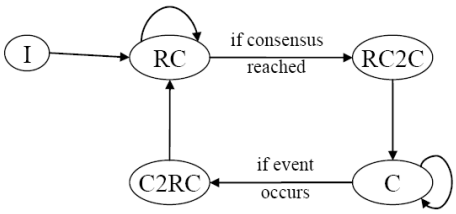

Fig. 1. Finite state machine of the node programming.

Castalia was designed right from the beginning to let the users implement/import easily their algorithms and protocols while making use of the features the simulator is providing. Proper modularization and a configurable, automated build procedure help towards this end.

V. Porting the Consensus algorithm to Castalia and Operation Modes of the Nodes

In order to implement the consensus algorithm of (2) using Castalia we have modified the application module to program the nodes to run the consensus algorithm. The nodes may be in three different operation modes denoted by I (initialization), RC (run_consensus) and C (consensus), and two additional transition operation modes, RC2C (run_consensus to consensus) and C2RC (consensus to run_consensus). The nodes execute a different program depending on the operation modes in which they are. Thus, figure 1 shows the evolution of each node regarding the operation modes. The actions taken by the node in each mode are described below:

- **I (initialization)** In this mode the node is initialized so that several macros and variables useful for our simulations are defined. The timers are also initialized. Each node defines a table to store the information sent by the neighbors, which is required to update the state according to 8 . After this initialization stage, the node takes a measurement of the physical parameter using the sensor module and sends an APP_NODE_STARTUP message to start the consensus algorithm.
- **RC(run_consensus):** In this mode the node is programmed to run the consensus algorithm and it also checks if consensus has been reached to commute to the C operation mode. The node acts in a different way if it receives a packet or has to transmit a packet.
  - When the node receives a new packet from a neighbor, the node updates the table with the new information included in the packet. Basically, the information encoded in the packet is the node identifier number, the state and the transmission power. Note that the table size does not increase with time since the old information of the neighbor is deleted.
  - When the node has to transmit a packet, the node computes its new state using (30) with the new information stored in its table. Afterwards, the packet is transmitted encoding the node identifier number, the new state and the transmission power. At this point, the node also checks if consensus is reached among its neighbors and itself. This is done comparing the new state with the previous one and with the state of the neighbors. If the node decides the consensus is reached, it will change its operation mode to RC2C. Otherwise, it will remain in the RC operation mode.
- **C(consensus):** In this mode the node spends most of the time in the sleeping mode but has to sense the environment from time to time, and wake up the network if it detects an abnormal value. The node also has to listen to its neighbors in order to switch its operation mode in case any neighbor detects an event. In the C operation mode the node reacts in a different way if it receives a packet or if it has a packet to transmit:
  - When the node receives a packet from a neighbor, it compares the state received from that neighbor with the last value stored in the table corresponding to that neighbor. If they are different, it means that an event might have occurred and the node changes to the C2RC transition operation mode.
  - When the node has to transmit a packet, it first takes a new measurement with the sensor module. This new measurement is compared to the last one toked on the last run of the consensus algorithm. If these values are sufficiently different, the node decides an event might have occurred and it changes to the C2RC transition operation mode.
- **RC2C (run-consensus to consensus)**. This is a transition mode in which the node switches from the RC operation mode to the C operation mode. In the RC2C operation mode, the node only changes the MAC parameters in order to reduce energy consumption.
- **C2RC(consensus to run-consensus):** This is a transition mode in which the node switches from the C operation mode to the RC operation mode. In the C2RC operation mode, the node changes the MAC parameters in order to alert and wake up the network. Actually we did not succeed in programming this part. So, the simulations that we will present, introduce only the other operation modes.

VI. Simulation Results

In this section the simulation results obtained by implementing the consensus algorithm of (3) with Castalia are presented.

A. Impact of the random topology with asymmetric links on the consensus value.

In this first set of simulations a network with 10 nodes is implemented. Each node initializes and runs the consensus algorithm, that is, the node is initially in the I operation mode and, after the initialization stage is completed, it switches to the RC operation mode and remains there. With this first set of simulations we want to check the impact of the randomness and time-variance of the network topology on the consensus value reached by the network. For that purpose we run two

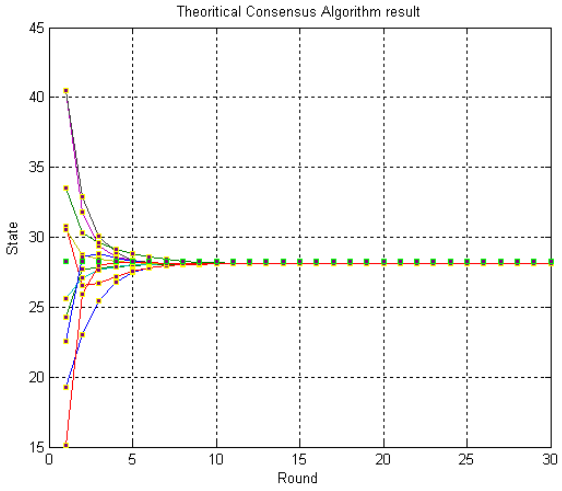

Fig. 2. Theoretical Consensus Algorithm for a network with $N = 10$ and $\varepsilon = 0.01$. The green squares show the average of the initial measurements

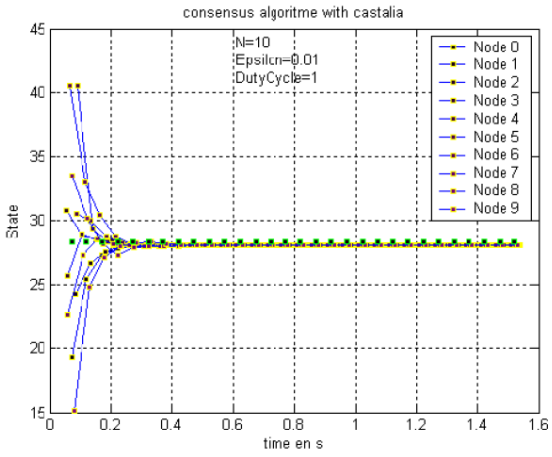

Fig. 3. Consensus algorithm with a network with $N = 10$ and $\varepsilon = 0.01$. The green squares show the average of the initial measurements.

different tests with $N = 10$ and $\varepsilon = 0.01$, and exactly the same initial measurements of the nodes, that is, with $\mathbf{x}_0$ equal in both tests. This is feasible because Castalia defines seeds for the different random number generators (RNG) that control different random processes. Then, by changing the corresponding seed we will have a different wireless channel and radio model which are reflected the environment conditions. In this case, we get a different results for different value of the simulations seeds. The parameters used in this first set of simulations are the following:

- Network with N=10 nodes.
- Initial measurements taken by the nodes: $\mathbf{x}_0$=[19.2355, 24.2819, 30.7719, 25.6261, 40.464, 30.5232, 40.4871, 22.5843, 33.5071, 15.141]
- Inter_packet_spacing=0.1s.
- MAC parameters: dutyCycle = 1, timeInterval = 1ms, RandomTxOffset = 1ms, backoffType = 1, backoffBaseValue = 16ms.
- Wireless channel parameters: PLd0 = $55dBm$, d0= 1m, sigma = 4, pathLossExponent = 2.4, allBidirectionalLinks =true, collisionModel = 2.
- Radio channel (CC2420 radio model [11]): dataRate = 250kbps, encodingType = NRZ, modulation Type = BPSK(QPSK), rxPower = 62mW, listenPower = 62mW, sleepPower = 1.4mW.

To understand the role of all these parameters you can look at Castalia user's manual [8].
With the same network topology we have run the equation 3 and we have obtained the figure 2 which report the theoretical simulation result. If we compare the value of the average consensus given by 2 and the other in figure 3 which obtained by our approach, we can say that they are similar. Moreover, it is important to print out that in figure 2 all nodes transmit and receive at the same time which is not practical in the reality. But, our method which introduces our simple random MAC is more realistic because it take in consideration the time and

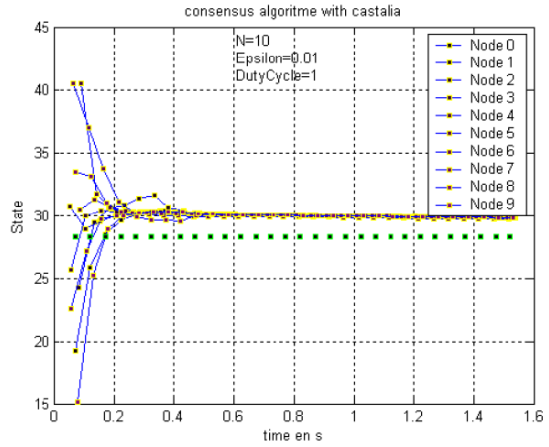

Fig. 4. Consensus algorithm with a network with $N = 10$ and $\varepsilon = 0.01$. The green squares show the average of the initial measurements.

collision notions.
Comparing the results shown in figures 3 and 4, it may be observed that the nodes reach a different consensus value in the two tests, although the initial set of measurements is exactly the same. This is due to the randomness of the channel and the MAC protocol which are controlled by different seeds (RNGs). Indeed, it may be observed that the initial state of the nodes is the same in both simulations. But, the time instants at which nodes update their states are different in the two simulations. This is why the consensus reached in the tests shown in figures 3 and 4 are different, and also different from the average consensus depicted with green squares.

B. Impact of ε on the consensus algorithm

In this section we want to evaluate with simulations the impact of the parameter ε on the consensus value reached by the network. The parameters used in this set of simulations are the same as the ones used in subsection A, the only change is affected the values of ε. The results are shown in figures 5, 6

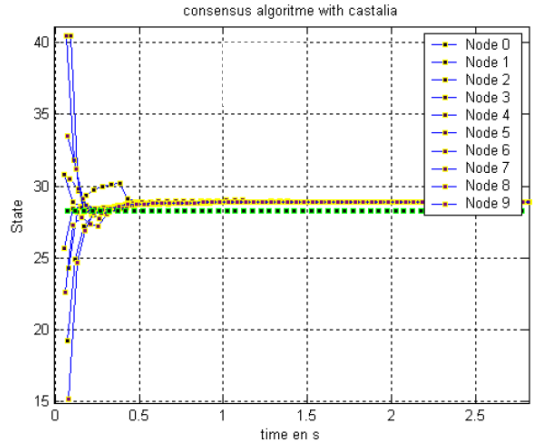

Fig. 5. Consensus algorithm with a network with $N = 10$ and $\varepsilon = 0.1$. The green squares show the average of the initial measurements.

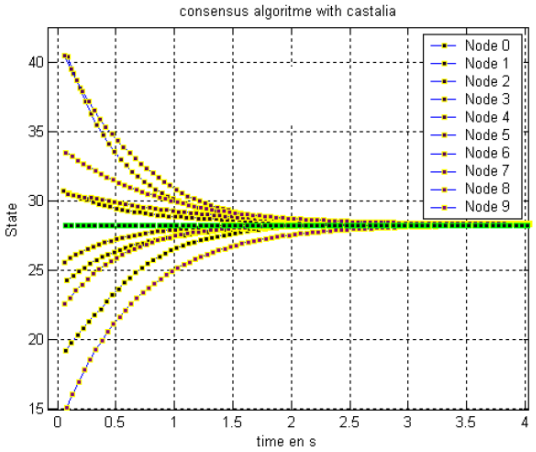

Fig. 6. Consensus algorithm with a network with $N = 10$ and $\varepsilon = 0.01$. The green squares show the average of the initial measurements.

and 7. As it is quite well-known, the network requires more time to reach the consensus as ε decreases. Indeed, as a rule of thumb the number of iterations increases at a rate of $1/\varepsilon$. Another effect we may observe in these simulations is that the variance of the consensus value reached by the network with respect to the average consensus value which is plotted with green squares, becomes smaller when ε increases. This result is justified theoretically in [11].

Table 8 presents some results showing the difference between the actual consensus value reached by the network and the average consensus, and how this difference depends on ε. This table shows results for a WSN composed of 4 nodes (by rows) and for four different tests with different values of ε (by columns). For each node the initial state and the final state at the iteration indicated in the corresponding column is included. As before, if ε decreases the iterative algorithm needs more iterations to converge but the consensus value is also closer to the average consensus, which is equal in this

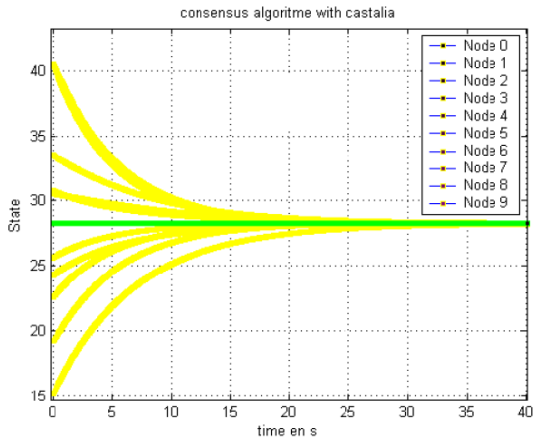

Fig. 7. Consensus algorithm with a network with $N = 10$ and $\varepsilon = 0.001$. The green squares show the average of the initial measurements.

		Iteration	epsilon	Iteration	epsilon	Iteration	epsilon	Iteration	epsilon
		10	0.1	50	0.1	500	0.01	5000	0.001
Node0	Initial state	19.7645		19.7645		19.7645		19.7645	
	Final state	**24.8831**		**24.9631**		**25.1418**		**25.1585**	
Node1	Initial state	24.4785		24.4785		24.4785		24.4785	
	Final state	**24.9656**		**24.9631**		**25.1418**		**25.1585**	
Node2	Initial state	30.7719		30.7719		30.7719		30.7719	
	Final state	**25.0365**		**24.9631**		**25.1418**		**25.1585**	
Node3	Initial state	25.6261		25.6261		25.6261		25.6261	
	Final state	**24.9803**		**24.9631**		**25.1418**		**25.1585**	

Fig. 8. Comparison of convergence rate and accuracy in the consensus reached by the network for a network with $N = 4$ nodes and different values of ε.

case to 25.16025.

C. Impact of power transmission on topology and consensus

In this section we check the impact of the power transmission on the topology and, therefore, on consensus with simulations. A WSN with N=9 and $\varepsilon = 0.01$ is considered. As before, after the nodes are initialized, they run de consensus algorithm and remain in the RC operation mode. The used parameters in this set of simulations are the following ones:

- Network with N=9 nodes.
- Inter_packet_spacing $= 0.1$s.
- MAC parameters: dutyCycle $= 1$, timeInterval $= 1$ms, RandomTxOffset $= 1$ms, backoffType $= 1$, backoffBaseValue $= 16$ms.
- Wireless channel parameters: PLd0 $= 55$dBm, d0 $= 1$m, sigma $= 4$, pathLossExponent $= 2.4$, allBidirectionalLinks =true.
- Radio channel (CC2420 radio model): dataRate $= 250kbps$, encodingType $=$ NRZ, modulation Type= BPSK(QPSK), rxPower $= 62$mW, listenPower= 62mW, sleepPower $= 1.4$mW.
- Initial measurements taken by the nodes: $\mathbf{x}(0)$= [14.2355, 15.0857, 21.7719, 17.6261, 29.6728, 39.5948, 15.4871, 24.5843, 18.5071].

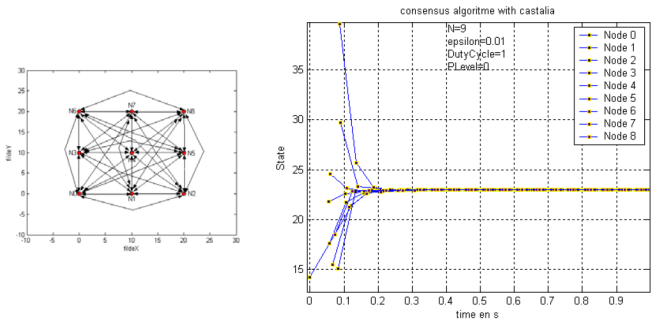

Fig. 9. Consensus algorithm with a network with $N = 9$, $\varepsilon = 0.1$ and power transmission level set to 0. Resulting connectivity map (left hand side) and temporal evolution of the state of the nodes (right hand side).

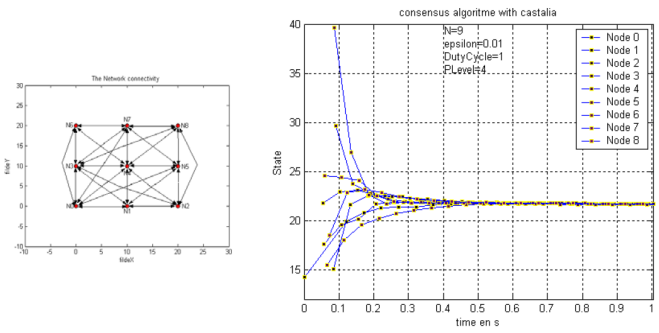

Fig. 10. Consensus algorithm with a network with $N = 9$, $\varepsilon = 0.1$ and power transmission level set to 4. Resulting connectivity map (left hand side) and temporal evolution of the state of the nodes (right hand side).

In Castalia, the nodes may transmit with 7 different levels of power, ordered from 0 to 6 where the 0 power transmission level is the maximum one. The simulation results for power transmission level 0, 4 and 6 are shown in the figures 9, 10 and 11, respectively. Two different figures are shown in each figure. The plot on the right hand side is the evolution with time of the node states and the plot on the left hand side is the resulting connectivity map so that an edge between two nodes is plot whenever there is a non zero probability of connection between those two nodes. The probability of connection for each pair of nodes is computed at simulation time.

Comparing the obtained results in the figures 9 and 10, in both cases the network is connected but yield to different convergence times. Indeed, the network with the largest power transmission level of 0 is full connected and, therefore, requires less time to converge compared to the network with the power transmission level of 4 which is connected but in a weaker sense. In the case of using the smallest power transmission level of 6 the network is disconnected and consensus will eventually be achieved by the 6 nodes that are connected as it may be seen in figure 11.

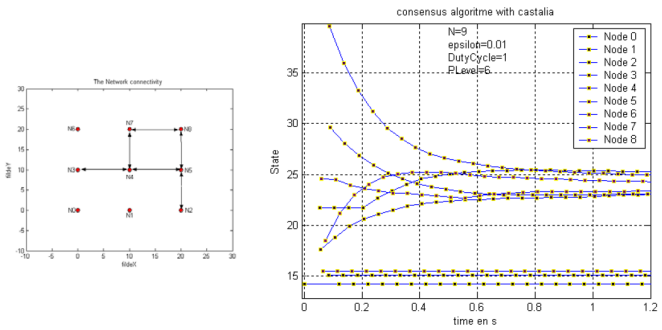

Fig. 11. Consensus algorithm with a network with $N = 9$, $\varepsilon = 0.1$ and power transmission level set to 6. Resulting connectivity map (left hand side) and temporal evolution of the state of the nodes (right hand side).

VII. Conclusion

In this paper a preliminary set of simulations obtained with Castalia implementing the consensus algorithm that converge on the state of equation 3 have been given. The objective of these simulations is to check the behavior of the algorithm using the realistic environment provided by the Castalia tool. We have seen that the network is able to operate using only local information without a preliminary organization of the nodes. However, the performance of this algorithm and protocol has to be checked in a much larger WSN and the parameters that control the MAC protocol have to be fixed to guarantee the required quality of the network from the application point of view.

References

[1] F. Akyidiz, Y. Sankarasubramaniam W. Su, and E. Cayirci, "A survey on sensor networks", IEEE Commun, August 2002.
[2] N. A. Lynch, "Distributed Algorithms, Morgan Kaufmann Publishers", Inc., 1997.
[3] S. Barbarosa , "Achienving Consensus in self-oganizing Wireless Sensor Networks: the impact of network topology on energy consumtion", In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP07), volume 2, pages II841II844, 15-20 April 2007.
[4] S. Kar and J.M.F. Moura. "Distributed average consensus in sensor networks with random link failures". In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP07), volume 2, pages II1013II1016, 15-20 April 2007.
[5] H. Pham, D. Pediaditakis, and A. Boulis, "From simulation to real deployments in WSN and back". In t2pWSN 2007, 2007.
[6] olfati saber and R. M. Murray, "Consensus Problems in Networks of Agents with Switching Topology and Time-Delays", IEEE Transactions on Automatic Control, 49(9):15201533, Sept. 2004.
[7] olfati saber, "Consensus and Cooperation in NetworkedMulti-Agent Systems", Proceedings of the IEEE, 95(1):215233, Jan. 2007.
[8] http://castalia.npc.nicta.com.au/
[9] L. Pescosolido, S. Barbarossa and G. Scutari "AVERAGE CONSENSUS ALGORITHMS ROBUST AGAINST CHANNEL NOISE" SPAWC 2008
[10] S. Silva Pereira and A. Pages Zamora. "Distributed average consensus in sensor networks with random topologies and asymmetric links". IEEE International Conference on Acoustics, Speech and Signal Processing, 2008.
[11] http://www-mtl.mit.edu/Courses/6.111/labkit/datasheets/CC2420.pdf