On the Distributed Binary Consensus Algorithm in Wireless Sensor Networks

Abdaoui Abderrazak* and Tarek Mohamed El Fouly†
*† College of Engineering Qatar University, Doha, Qatar PoBox 2713
Email: abdaoui@ieee.org
* Imperial College London, SW7 2AZ, UK

ABSTRACT

We consider the binary consensus problem over the wireless sensor networks (WSN), where each node of the network initially observes one of two states and the aim of each node is to decide which one of the two states was held by the majority of nodes. In this paper we consider the averaging of a measurement in a WSN composed of N nodes. We evaluate the distributed binary consensus algorithm by deriving the average convergence time of the algorithm. Since this time has been evaluated theoretically for mathematical aspects [1], we complete this work by considering real life environment including packet loss phenomenon and automatic repeat request (ARQ) protocol used in real message transmissions. In addition we compare the new analytical results with those obtained by a hardware emulation of the same binary consensus algorithm under TinyOS and TOSSIM. In performance evaluation, we consider the analysis of the average convergence time for node states. In the simulations, we apply the distributed binary consensus algorithm for fully connected, ring, path, Erdos Reny random, and star-shaped topologies.

Keywords—Binary consensus algorithm, distributed computation, TinyOS, IRIS Motes, Tossim, Wireless sensor network, ARQ, packet loss.

I. INTRODUCTION

Distributed computations within wireless sensor networks (WSNs) are currently of great interest to engineers and researchers. The significant challenge in this field is how can we achieve the overall reliability of the whole network in face of faulty nodes. The consensus problem is related to the distributed manipulation of a single data within the nodes of the network [2]–[4]. In order to accomplish this, it is often required for all nodes in the network to reach agreement on a specific piece of data required for a computation. In [2], Bashir *et al.* use consensus algorithm to provide a powerful solution for distributed routing failure detection in wireless sensor networks. Indeed, a consensus about suspected node is generated by the collaboration of neighbors node. A simple algorithm for consensus is presented; every neighbor considers a decision factor for every other neighbor in order to generate a unified agreement about the node under suspicion. In their work, the authors analyze and show that their approach performs better for energy conservation and node lifetime better than previously. However, this method is limited to a network with a specific tree around the suspected node, and mechanism is more close to the well known voting method [5]. In [3], the authors proposed a new average consensus algorithm, where each node selects its own weights on the basis of some local information about its neighborhood. The proposed algorithm is tailored for networks having cluster structure. The neighborhood algorithm is designed to identify such links and give them higher weights in order to speed-up information propagation among different parts of the networks. In realistic sensor network topologies, the algorithm shows faster convergence than other existing consensus protocols [5]–[8]. In their work in [4], the authors present an iterative decentralized consensus algorithm for routing in WSN by considering the minimization of the number of iterations and then ensure the limitation of the energy consumption.

Binary consensus algorithm is a subclass of consensus problem and it is applied when there is delimitation on the memory and the processing speed [1], [9]–[20]. In a binary consensus algorithm, all nodes initially compute a TRUE or FALSE answer to a given question (such as whether the current temperature is over 35 degrees) and then attempt to reach agreement on which state the majority of nodes hold. In [9], Mostefaoui *et al.* proposed a reduced complexity algorithm in asynchronous systems with crash failures. In their algorithm, each process runs a series of binary consensus subroutines, sequentially, in order to solve multivalued consensus. Nevertheless, the number of subroutines necessary to solve one multivariate consensus instance is unlimited and depends on the message delay. Applications of this algorithm include coordination of autonomous agents, estimation, and distributed data fusion in *ad-hoc* or social networks. In [10], the authors consider the binary consensus problem where each node in the network initially possesses one of two states and the goal is to decide which one of both states was initially held by the majority of nodes. The states considered in their paper is 0 (when the measurement is under 0.5), 1 (when the measurement is over 0.5) and e (for undetermined measurement). The authors show also that extending, both the signaling and memory, by just one state, may improve the reliability and speed to reach the correct consensus. Thus, it has been proved that the probability of error decays exponentially with the number of nodes N.

In [11], the authors design quantized average consensus algorithm on arbitrary connected networks. In fact, quantized algorithms cannot produce a real, analog average. Instead, their algorithm reaches consensus on the quantized interval that contains the average. In their work, the authors prove that this consensus in reached in finite time almost surely. In [12], the authors develop and implement a randomized consensus protocol for the contexts where users can frequently remain dispersed over a large area, frequently resulting in sparse patches containing fewer users per unit area. Their work establishes the liveness condition that any ad-hoc network must satisfy for consensus to be solvable using the randomized approach. Performance analysis of the developed protocol show the effects of randomization in speeding up the consensus when network gets sparser. In [13], the authors consider a network that is trying to reach consensus over the occurrence of an event while communicating over additive white Gaussian noise (AWGN) channels. Indeed, the authors characterize the impact of different link qualities and network connectivity on consensus performance.

In [1], the authors derived an upper-bound of the expected convergence time of the distributed binary consensus algorithm. The bound is derived for a particular network topology with a connected graph. In addition, they instantiated the upper bound for some other network topologies such as complete graph, star-shaped, ring and Erdos-Renyi random graphs. However, the contribution of Draief *et al.* presents the average convergence time without considering real conditions of the WSN such as communication protocols, routing, ACK and NOACK. Indeed, the average convergence time depends on the implementation and tests conditions of the binary consensus algorithm over the simulation and testbed framework.

In this paper, we extend the contribution of Draief *et al.* [1] by the deployment of distributed binary consensus algorithm in a wireless sensor network. We propose the implementation of the binary consensus algorithm, where each node has its own temperature value at the beginning and all the nodes start communicating with each other to reach the final decision. In this system, the final decision represents the majority opinions of all the nodes. Moreover, we implement our system on TinyOS, and simulate it using a simulator called TOSSIM [21]–[25].

The remainder of the paper is organized as follows. In Section II, we present the binary consensus algorithm. Section III details the distributed implementation and the routing features of the binary consensus algorithm. The analytical characterisation of the average convergence time of the binary consensus algorithm is detailed in Section IV. Section V-A presents the testbed environment and the simulation tools based on TinyOS operating system. The simulations as well as the hardware results followed by analysis are presented at Section V. In the final Section, we conclude the paper.

II. The binary Consensus Algorithm

Recently, there has been a great deal of research on distributed computing with respect to the processing, storage constraints and the communication channel between nodes. For example sensors, tags, PDAs and wireless communications have become massively pervasive in houses, streets and in offices and they require capability of dynamically organize and adapt their computational/communication/storage capabilities. We detail the interval binary consensus algorithm with the routing protocol used in the simulation and the testbed of the wireless sensor network. Assume that at every time instant, each node of the network has a state taking as values 0, e_0, e_1 and 1. Consider that the states satisfy $0 < e_0 < e_1 < 1$. Here, e_0 represents the values smaller than $\frac{1}{2}$ and e_1 refers to the values larger than $\frac{1}{2}$. As an example, we consider a network with four nodes labeled n_1, n_2, n_3 and n_4, starting in state $(0, 0, 1, 0)$. A first interaction between node n_3 and node n_4 with opposite states, the two nodes disagree then they will become undecided and their new states will be respectively e_0 and e_1. The new vector of states becomes $(0, 0, e_0, e_1)$. If this interaction happens between node n_1 and node n_2 then no change in their states and the vector of states remains unchanged. If node n_3 and node n_4 interact again, their states are swapped according to the update rules and the vector of states will be $(0, 0, e_1, e_0)$.

A. Update rules

In distributed binary consensus algorithm, when two nodes get in contact, the states at each node are updated according to the following table

Actual states (node1 , node2)		New states (node1 , node2)
$(0\ ,\ 1)$	$\longmapsto$	$(e_0\ ,\ e_1)$
$(e_0\ ,\ 1)$	$\longmapsto$	$(1\ ,\ e_1)$
$(e_1\ ,\ 0)$	$\longmapsto$	$(0\ ,\ e_0)$
$(e_0\ ,\ 0)$	$\longmapsto$	$(0\ ,\ e_0)$
$(e_1\ ,\ 1)$	$\longmapsto$	$(1\ ,\ e_1)$
$(e_0\ ,\ e_1)$	$\longmapsto$	$(e_1\ ,\ e_0)$
$(e_x\ ,\ e_y)$	$\longmapsto$	$(e_x\ ,\ e_y)$

TABLE I. Update rules of the states for two communicating nodes.

B. Convergence of the algorithm

As indicated in [10], the communications between nodes are assumed to be asynchronous where the instance of interaction is Poisson with rate $q_{ij} \geq 0$. If we denote by $V = \{n_1, n_2, \ldots, n_N\}$ the set of nodes, the interaction rates are given by the matrix $\mathrm{Q} = (q_{ij})\,, \quad i, j \in V\ \mathrm{G} = (V, \mathrm{E})$ comprising a set V of vertices or nodes together with a set E of edges or lines representing the set of couples $(i, j),\ \forall i, j \in V$. The graph $\mathrm{G} = (V, \mathrm{E})$ is induced by the matrix Q with $(i, j) \in \mathrm{E}$ if only if $q_{ij} > 0$. Let us characterize the average convergence time of the algorithm according to [1] in a real-life environment. As the algorithm runs, it goes through following phases

- Phase 1 : (disappearance of state 1) starts from initial state and ends when nodes in state 1 disappear upon interacting with nodes in state 0.
- Phase 2 : (disappearance of state e_1) follows the first phase and ends when state e_1 disappears. Only nodes with state e_0 and state 0 remain.

- At the end of phase 1, none of the nodes is in state 1 $(2\alpha - 1)N$ are in state 0 and the remaining $(2(1-\alpha)N)$ nodes are in either state e_0 or state e_1. $\alpha > 1/2$ denotes the fraction of nodes that initially held the majority state.

- At the end of phase 2, there are exactly $(2\alpha - 1)N$ nodes in phase 0 and $2(1-\alpha)N$ nodes in state e_0. According to the work in [1], the smallest time at which all the nodes in state 1 are depleted is given by its mean

$$\mathbb{E}\left(T_1\right) \leqslant \frac{1}{\delta(\mathrm{Q}, \alpha)}\left(\log(N) + 1\right), \tag{1}$$

where $\mathbb{E}(x)$ denotes the mean of x and $\delta = \min|\lambda_{\mathrm{Q}_S}|_{S \subset V, |S| = (2\alpha-1)N}$ with λ_{Q_S} is the largest eigenvalue of Q_S. Q_S is derived from the contact rate matrix Q as follows

$$q^S_{i,j} = \begin{cases} -\sum_{l\in V} q_{i,l} & i = j \\ q_{i,j} & i \in S^c, \quad j \neq i \\ 0, & i \in S, \quad j \neq i \end{cases} \tag{2}$$

Where S is a subset of the set of vertices V and $S^c = V \setminus S$. Furthermore, the time T_2 for all the nodes in state e_1 to be depleted starting from initial state with no node in state 1 will have the following average

$$\mathbb{E}(T_2) \leqslant \frac{1}{\delta(\mathrm{Q}, \alpha)} (\log(N) + 1), \tag{3}$$

1) Case of N-vertices complete graph network : A network with a complete graph, $N > 1$ nodes and edge $e \in \mathrm{E}$ and a rate $q = \frac{1}{N-1}$ $\delta(\mathrm{Q}, \alpha) \geqslant (2\alpha - 1)$, then for every fixed $\alpha \in (1/2, 1]$, the expected convergence time during the first phase is given by

$$\mathbb{E}(T_1) = \frac{1}{2\alpha - 1} \log(N) \;+\; O(1) \tag{4}$$

the approximation function $o(1)$ is defined as $E(T_1)$ goes to ∞ $\frac{1}{2\alpha-1}\log(N) + const$ when N goes to ∞.

When α approaches $1/2$, case where initially there is an equal number of nodes in states 0 and state 1, according to [1], $E(T_1)$ is given by

$$\mathbb{E}(T_1) = \frac{\pi^2}{6} N(1 + o(1)), \tag{5}$$

where $o(1)$ is the approximation function that goes to 0 when N go to ∞.

2) Case of N-vertices paths network : A network with a path graph is characterized by a sequence of vertices such that from each of its vertices, there is an edge to the next vertex. We consider a path with $N > 1$ nodes, where each edge is activated at a rate $q = 1$ at instances of a Poisson process. The contact rate matrix is given by $q_{i,i+1}$ for $i = 1, \dots, N-1$ and $q_{i,i-1}$ for $i = 2, 3, \dots, N$ and all other elements equal to 0.

$$\mathrm{Q} = \begin{pmatrix} 0 & 1 & 0 & & \dots & 0 \\ 1 & 0 & 1 & 0 & & \vdots \\ 0 & 1 & 0 & 1 & \ddots & \\ & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 0 & 1 \\ 0 & \dots & & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

The average convergence time for $N > 1$ and $\alpha \in [1/2, 1)$ and for each phase $l = 1$ and $l = 2$

$$\mathbb{E}(T_l) \leqslant \frac{16(1-\alpha)^2}{\pi^2} N^2 \log(N) \;+\; O(1) \tag{7}$$

3) Case of N-vertices cycle networks : A network with a ring graph is characterized by a graph of $N > 1$ nodes which contains a single cycle. In this graph, each edge is activated at instances of Poisson process with rate 1. The contact rate is given by $q_{i,i+1} = 1$ for $i = 1, \dots, N-1$, $q_{i,i-1} = 1$ for $i = 2, 3, \dots, N$ and $q_{N,1} = q_{1,N} = 1$ for all other elements.

$$\mathrm{Q} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 1 \\ 1 & 0 & 1 & 0 & \dots & 0 \\ 0 & 1 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & 0 & 1 & 0 & 1 \\ 1 & 0 & \dots & 0 & 1 & 0 \end{pmatrix} \tag{8}$$

The average convergence time for $N > 1$ and $\alpha \in [1/2, 1)$ and for each phase $l = 1$ and $l = 2$

$$\mathbb{E}(T_l) \leqslant \frac{4(1-\alpha)^2}{\pi^2} N^2 \log(N) \;+\; O(1) \tag{9}$$

4) Case of N-vertices star-shaped network : This kind of network has a complete bipartite graph with an internal node (called also hub) and $N - 1$ leaves. The contacts between a leaf node and the hub are assumed to occur at instances of Poisson process with rate $\frac{1}{N-1}$. The elements of matrix Q are given by $q_{1,i} = q_{i,1} = \frac{1}{N-1}$ for $i = 2, \dots, N$, and 0 for the other, Q is defined as follows

$$\mathrm{Q} = \begin{pmatrix} 0 & q & \dots & q \\ q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ q & 0 & \dots & 0 \end{pmatrix}, \tag{10}$$

the average convergence time for phase l satisfies

$$\mathbb{E}(T_l) \leqslant \frac{1}{2\alpha - 1} N (\log(N) + 1), \tag{11}$$

and

According to Draief [1], for $N > 1$ and $\alpha \in [1/2, 1)$ the expected time to deplete nodes in state 1 satisfies

$$\mathbb{E}(T_1) = \frac{1}{(2\alpha-1)(3-2\alpha)} N \log(N) \;+\; O(n). \tag{12}$$

where $O(n)$ is approximated by Kn when n goes to ∞, with K a positive constant.

III. Distributed implementation and Routing Protocols

In this section, we study the use of the binary consensus algorithm in wireless sensor network real-life. We also detail, the routing protocols and their implementation for a wireless sensor network characterized by fully connected, paths, ring, star-shaped and Erdos-Rényi Random Graphs, respectively. In general, regardless to the topology of the network, let us consider two nodes called respectively *"Localnode"* and *"Othernode"* getting connected. If we assume that *"Localnode"* receives the state of *"Othernode"*, the distributed version of the binary consensus algorithm, running in both *Localnode* and *"Othernode"*, is described by the pseudo-code in Algorithm 1. This pseudo code applies exactly the Binary consensus rules defined in II-A with communication features and in a distributed manner.

In the following section, we consider the implementation of the distributed binary consensus algorithm with an emphasis on

Algorithm 1 Distributed Binary Consensus Algorithm

```
 1: if  node_state = 0 and othernode_state = 1  then
 2:     Send 0 $\rightsquigarrow$ OtherNode
 3:     node_state $\leftarrow \mathbf{e_1}$
 4: else if  node_state = 1 and othernode_state = 0  then
 5:     Send 1 $\rightsquigarrow$ OtherNode
 6:     node_state $\leftarrow \mathbf{e_0}$
 7: else if  node_state = $\mathbf{e_0}$ and othernode_state = 0  then
 8:     Send $\mathbf{e_0}$ $\rightsquigarrow$ OtherNode
 9:     node_state $\leftarrow \mathbf{0}$
10: else if  node_state = $\mathbf{0}$ and othernode_state = $\mathbf{e_0}$  then
11:     Send $\mathbf{0}$ $\rightsquigarrow$ OtherNode
12:     node_state $\leftarrow \mathbf{e_0}$
13: else if  node_state = $\mathbf{e_0}$ and othernode_state = $\mathbf{e_1}$  then
14:     Send $\mathbf{e_0}$ $\rightsquigarrow$ OtherNode
15:     node_state $\leftarrow \mathbf{e_1}$
16: else if  node_state = $\mathbf{e_1}$ and othernode_state = $\mathbf{e_0}$  then
17:     Send $\mathbf{e_1}$ $\rightsquigarrow$ OtherNode
18:     node_state $\leftarrow \mathbf{e_0}$
19: else if  node_state = $\mathbf{e_0}$ and othernode_state = $\mathbf{1}$  then
20:     Send $\mathbf{e_0}$ $\rightsquigarrow$ OtherNode
21:     node_state $\leftarrow \mathbf{1}$
22: else if  node_state = $\mathbf{1}$ and othernode_state = $\mathbf{e_0}$  then
23:     Send $\mathbf{1}$ $\rightsquigarrow$ OtherNode
24:     node_state $\leftarrow \mathbf{e_1}$
25: else if  node_state = $\mathbf{e_1}$ and othernode_state = $\mathbf{1}$  then
26:     Send $\mathbf{e_1}$ $\rightsquigarrow$ OtherNode
27:     node_state $\leftarrow \mathbf{1}$
28: else if  node_state = $\mathbf{1}$ and othernode_state = $\mathbf{e_1}$  then
29:     Send $\mathbf{1}$ $\rightsquigarrow$ OtherNode
30:     node_state $\leftarrow \mathbf{e_1}$
31: else if  node_state = $\mathbf{0}$ and othernode_state = $\mathbf{e_1}$  then
32:     Send $\mathbf{0}$ $\rightsquigarrow$ OtherNode
33:     node_state $\leftarrow \mathbf{0}$
34: else if  node_state = $\mathbf{e_1}$ and othernode_state = $\mathbf{0}$  then
35:     Send $\mathbf{e_1}$ $\rightsquigarrow$ OtherNode
36:     node_state $\leftarrow \mathbf{e_0}$
37: else
38:     othernode_state=DEAD
39: end if
```

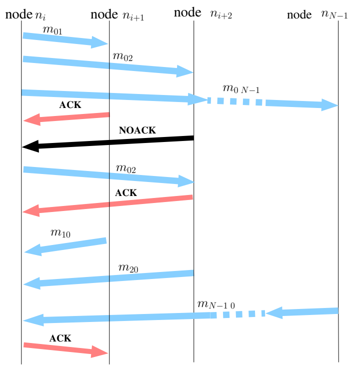


Fig. 1. Protocol for Star shaped topology

the routing features for the topologies introduced previously. In addition, we detail the protocol for each graph such as star-shaped, fully connected and ring graph topology. We assume here that the network is composed of N nodes. For

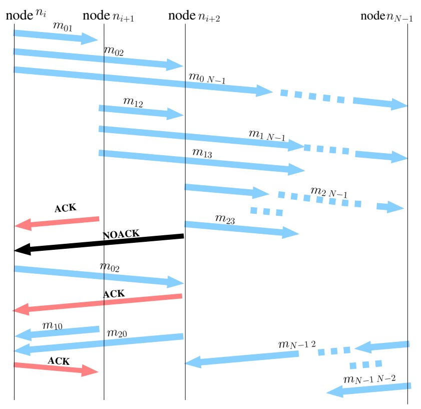


Fig. 2. Protocol for fully connected topology.

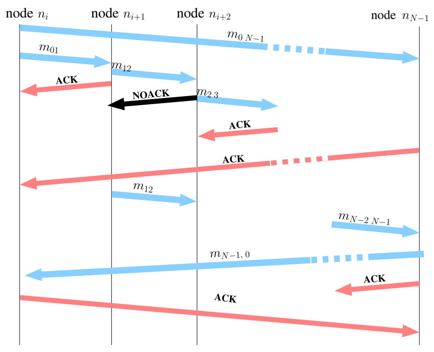


Fig. 3. Protocol for cycle topology.

both simulation and hardware implementation, the algorithm running at each node starts by an initialization step. This step starts when the sensor mote is turning on. The initial value is set when the *Boot.booted* event is signaled. For the simulation, the initialization is also done when *Boot.booted* occurs and the measured value is taken as random.

A. Protocol and binary consensus algorithm for star-shaped topology

For this topology, all nodes in the network are connected to a central node (assumed to be node N_0) and no others. Each node receives a message $m_{0,i}$ from node 0 with $i \in \{1, \ldots, N\}$ containing a state from $\{0\ , e_0\ , e_1$ and $1\}$. After applying local version of the binary consensus algorithm, the received node retransmits a message to node 0 containing the states decided by the algorithm. The details of this protocol and the acknowledgement mechanism are shown in details in Fig. 1. In addition, an acknowledgment mechanism is employed to resubmit each non received message after a small timeout optimized according to the network charge.

B. Protocol and binary consensus algorithm for fully connected topology

For this topology, each node exchanges messages with all other nodes in the network. The network is scanned when each node with index n_i communicates with all nodes with higher index such as $n_{i+1}, \ldots, n_{i+N}$. Then, in each receiving node, a receiving event calls a subroutine containing the update rules of the binary consensus algorithm and then exchanges states, if necessary, with the submitting node. Once the source node detects the receiving events, the received message is read and analyzed to collect the *OtherState*, which represents the state of the corresponding node, and then apply the update rules of the binary consensus algorithm as in the pseudo-code of algorithm 0. Fig. 2 presents the details of the messages exchanged by the nodes in fully connected topology. An acknowledgment mechanism is also employed to resubmit each non received message after a small timeout optimized according to the network charge.

C. Protocol and binary consensus algorithm for ring topology

For this topology, each node exchange message with node that follows. Specifically, a node with index n_i communicates with the node indexed n_{i+1} and then the receiving node applies the update rules of the binary consensus algorithm and sends its state, if necessary, to the source node n_i. At the end of the ring, node n_0 communicates with node n_N and applies the update rules of the binary consensus algorithm. The process is repeated until convergence is obtained. As in the other topologies, an acknowledgment mechanism is also employed to resubmit each non received message after a small timeout optimized according to the network charge. Fig. 3 presents the protocol for ring topology.

IV. Average convergence time of the BCA in real WSN environment

The major problem in wireless sensor network and distributed algorithms is how to introduce the wireless losses and packet losses effects in overall performance analysis. In the previous work, we have characterized the average convergence time of the binary consensus algorithm (BCA) on WSN for several topologies. In this part, we extend the analytical derivation of the average convergence time of the BCA in [1] to cover the real life conditions of the wireless channel such as packet losses and congestions.

A. Packet losses and wireless sensor network

Packet delivery performance is of great importance in wireless sensor network, since it reflects the lifetime of the network. Packet delivery or packet loss is mostly depending on the environment such as channel noise, interference and congestion due to multiple reception from other sources. In WSN performance analysis there exist several studies to evaluate the packet loss/delivery. Woo et al. [26] has examined the packet loss problem between a pair of nodes. The variation of packet loss is specific to the physical environment and the congestions due to multiple receptions. This packet loss increases with distance relating the base station and the sensor node and the noise/interference coming from external sources. Packet loss is the failure of one or more transmitted packets to arrive at their destination. This event can cause noticeable effects in all types communications; Errors in Data, create Jitter in video conference, in worst case, packet loss can cause sever mutilation of received data and finally broken images or complete absence of a received signal.

B. System Modelization

Let $N^i(p_d)$, be the minimum number of transmissions per processed packet at each node i that guarantee a successful delivery to the neighbor node with probability p_d. Let also, $E(p_d)$ be the expected value of the number of transmissions (data and ACKs) per processed data to guarantee a successfully delivery with probability p_d.

Proposition 1: For single hop transmission, the minimum number of transmissions per processed packet is explained as follows:

$$N^{ij}(p_d) = \left\lceil \frac{\log(1-p_d)}{\log(p_{ij})} \right\rceil \tag{13}$$

Proof : A processed message transmitted by node i is received by node j successfully with probability $1 - p_{i,j}$ regardless the ACK mechanism. If no ACK received within a predetermined timeout, the message is retransmitted again. For any given number of transmissions per sensed data N^{ij}, the message is delivered to node j successfully with probability $1 - p_{ij}^{N^{ij}}$. For a good reliability along the transmission, we require that each N^{ij} value satisfies $(1 - p_{ij}^{N^{ij}}) \geq p_d$. It is straightforward to verify that for equality (minimum value of N^{ij}), we obtain a proof of (13) We evaluate now, the total expected number of transmission (data and ACKs) $E(p_d)$ per processed packet.

Proposition 2 : For ARQ with Acknowledgment protocol,

$$E(p_d) = \frac{1 - (1 - \overline{p}_{ij}\overline{q}_{ij})^{N^{ij}(p_d)}}{\overline{p}_{ij}\overline{q}_{ij}}(1 + \overline{p}_{ij}) \tag{14}$$

Let X^{ij} be the number of processed packet transmissions from node i to node j (one single hop), and Y^{ij} be the number of ACKs from node j to node i. Note here that X^{ij} is a truncated geometrically distributed random variable with success probability of $\overline{p}_{ij}\overline{q}_{ij}$ taking values inf $\{1, \ldots, N^{ij}(p_d)\}$. Since ACK is sent for each packet that is successfully received at node i, then

$$E[Y^{ij}] = \overline{p}_{ij} E[X^{ij}] \tag{15}$$

for one hop communication, $E(p_d)$ is given by

$$E(p_d) = E[X^{ij} + Y^{ij}] = \frac{1 - (1 - \overline{p}_{ij}\overline{q}_{ij})^{N^{ij}(p_d)}}{\overline{p}_{ij}\overline{q}_{ij}}(1 + \overline{p}_{ij}) \tag{16}$$

V. Simulation Results

A. Test Environment

In this work, we consider the simulation of the distributed binary consensus interval in WSN composed of N sensor nodes. We analyzed the average convergence time of the binary consensus algorithm with N taking values up to 1000 nodes. In addition, we show the complex behavior of the processing and communication protocols between nodes. There are a number of options for testing a protocol for WSNs, such as a WSN simulator (TOSSIM), micro-controller instruction set simulators (AVRORA) and testbeds (MoteLab). In our approach, we selected a test method that allows us to use the same test case and source code (in nesC). For simulation, we selected TOSSIM simulator which is given as part of module in TinyOS distribution.

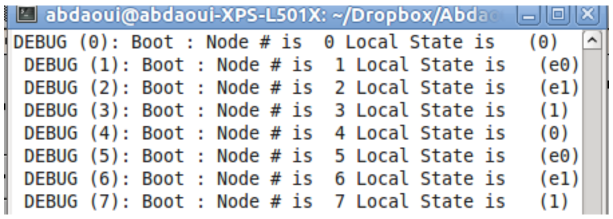
TinyOS : TinyOS is an open source operating system specifically designed for wireless sensor motes. This OS has several important features that influence nesC's design : a simple event based concurrency model and split phase operations.

TOSSIM simulator : TOSSIM is a discrete event simulator for TinyOS applications which replaces components with simulation implementations. As TOSSIM runs on a PC, users can examine TinyOS code using debuggers and other tools. The worst limitation of TOSSIM is that each node in the network uses the same nesC code as the others. This means that all nodes are identical which is not correct in general. TOSSIM can also use a large simulated radio topology, but limited memory forces us to use low charged network. Since TOSSIM is the unique simulator in WSN and because the application at each node is an instantiation of the distributed binary consensus algorithm, we use this simulation tool in a network with a number of nodes under 1000.

In this Section, we evaluate the distributed binary interval consensus algorithm using TOSSIM. To do that, let us describe the requirements and properties of the distributed binary consensus program for measurement averaging. The program is written in the nesC language under TinyOS operating system. In terms of running the simulation experiments, a small script written in python or C++ is used to control and manage the overall simulation with TOSSIM. The script is used to configure the network topology, start up the nodes, etc. The source code being simulated contains large numbers of debug messages which can be analyzed to extract routing information and control packets during simulation.

B. Simulation Behavior

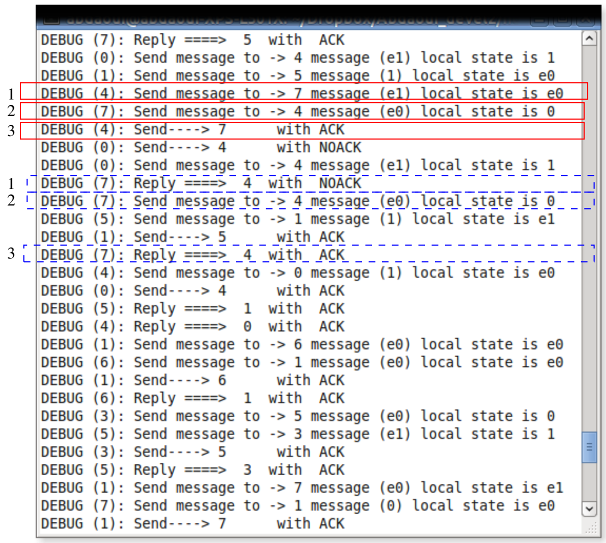
We consider the test of the distributed binary consensus algorithm for a fully connected wireless sensor network equipped with N nodes. We note here that the fully connected topology is based on a complete graph with $N(N-1)/2$ edges. The screen shot presented by Figs. 4 and 5 show the debugging results for the booting and processing steps respectively. As given by the booting step in Fig. 4, the initial values of node states are taken randomly from the set $\{0, e_0, e_1, 1\}$ and the vector states representing the nodes states is $\{0, e_0, e_1, 1, 0, e_0, e_1, 1\}$.



```
abdaoui@abdaoui-XPS-L501X: ~/Dropbox/Abda
DEBUG (0): Boot : Node # is  0 Local State is   (0)
 DEBUG (1): Boot : Node # is  1 Local State is   (e0)
 DEBUG (2): Boot : Node # is  2 Local State is   (e1)
 DEBUG (3): Boot : Node # is  3 Local State is   (1)
 DEBUG (4): Boot : Node # is  4 Local State is   (0)
 DEBUG (5): Boot : Node # is  5 Local State is   (e0)
 DEBUG (6): Boot : Node # is  6 Local State is   (e1)
 DEBUG (7): Boot : Node # is  7 Local State is   (1)
```

Fig. 4. Messages exchanged for fully connected topology: booting step.

After the boot step, the communication is initiated by the node with lower index and transmits its state to all the nodes with higher index. Each receiving node applies the update rules of the local binary consensus algorithm and replies to the source node by sending its local state and modifying its state according to the update rules. The fully connected routing table is scanned after all pairs of sends and replies. An acknowledge mechanism is employed to check and resubmit each lost packet if necessary. In Fig. 5, the solid and dashed rectangles track the connections established between node 4 and node 7 and the node states at each step. In debugging steps, indicated by the red rectangle number 1, node 4 with state e_0 send a message e_1 to node 7. After rules update, the result is the connection established by node 7, with new state 0, to send the state e_0 to the node number 4. The third solid rectangle presents the ACK of the first transmission from node 4 to node 7. However, the dashed rectangle number 1 shows a NOACK for the responding connection, from node 7 to node 4. In dashed rectangle number 2, a retransmission of the missed message is done after the last NOACK. The dashed rectangle number 3 shows the transmission with ACK of the last missed message from 7 to 4.



```
  DEBUG (7): Reply ====>  5  with  ACK
  DEBUG (0): Send message to -> 4 message (e1) local state is 1
  DEBUG (1): Send message to -> 5 message (1) local state is e0
1 DEBUG (4): Send message to -> 7 message (e1) local state is e0
2 DEBUG (7): Send message to -> 4 message (e0) local state is 0
3 DEBUG (4): Send----> 7      with ACK
  DEBUG (0): Send----> 4      with NOACK
  DEBUG (0): Send message to -> 4 message (e1) local state is 1
1 DEBUG (7): Reply ====>  4  with  NOACK
2 DEBUG (7): Send message to -> 4 message (e0) local state is 0
  DEBUG (5): Send message to -> 1 message (1) local state is e1
  DEBUG (1): Send----> 5      with ACK
3 DEBUG (7): Reply ====>  4  with  ACK
  DEBUG (4): Send message to -> 0 message (1) local state is e0
  DEBUG (0): Send----> 4      with ACK
  DEBUG (5): Reply ====>  1  with  ACK
  DEBUG (4): Reply ====>  0  with  ACK
  DEBUG (1): Send message to -> 6 message (e0) local state is e0
  DEBUG (6): Send message to -> 1 message (e0) local state is e0
  DEBUG (1): Send----> 6      with ACK
  DEBUG (6): Reply ====>  1  with  ACK
  DEBUG (3): Send message to -> 5 message (e0) local state is 0
  DEBUG (5): Send message to -> 3 message (e1) local state is 1
  DEBUG (3): Send----> 5      with ACK
  DEBUG (5): Reply ====>  3  with  ACK
  DEBUG (1): Send message to -> 7 message (e0) local state is e1
  DEBUG (7): Send message to -> 1 message (0) local state is e0
  DEBUG (1): Send----> 7      with ACK
```

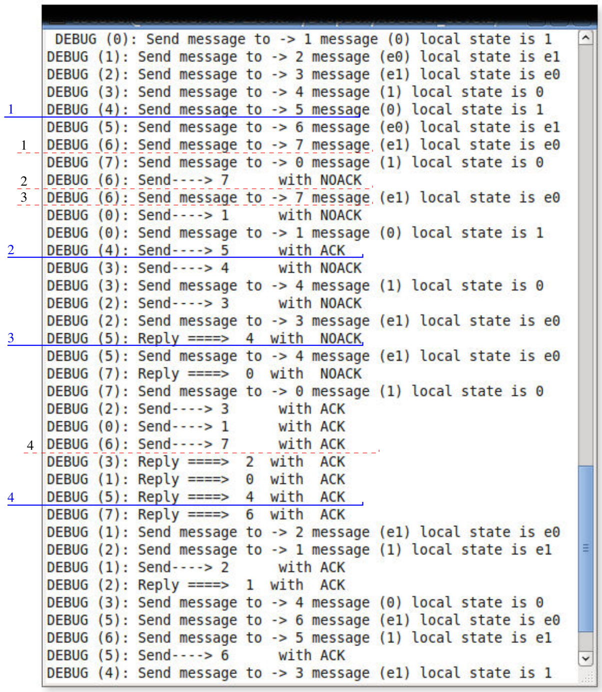
Fig. 5. Messages exchanged for fully connected topology.

Without loss of generalities, in Fig. 6, we show the simulated wireless sensor network with N=8 nodes and ring graph topology. In this network, the communication is established in cycle as $0 \rightarrow 1$, $1 \rightarrow 2$, $2 \rightarrow 3 \ldots$, $6 \rightarrow 7$ and $7 \rightarrow 0$. Briefly, from Fig. 6, we can summarize with the solid line and the dashed line that the connection between 4 and 5 and that between 6 and 7 succeed after one NOACK.

C. Simulation Results

In the following, we consider the evaluation of the average convergence time of the distributed binary consensus algorithm in a wireless sensor network equipped with N nodes. In this simulation, we provide a comparison between the analytical results obtained in [1] and the simulated results obtained by the TOSSIM simulator. The results are given for three topologies: fully connected, star, and ring.

Fig. 7 shows our results for a star topology. The blue lines represent analytical results from [1], while the red points represent the results of our TinyOS simulation. According to these results, the mean convergence time is in inverse ratio with $2\alpha - 1$ but increase when the number of nodes of the network growth. The particular value of $\alpha = 0.5$ means that there is no group of nodes that initially held the majority states. All the initial states are spread on all the nodes of



```
   DEBUG (0): Send message to -> 1 message (0) local state is 1
  DEBUG (1): Send message to -> 2 message (e0) local state is e1
  DEBUG (2): Send message to -> 3 message (e1) local state is e0
  DEBUG (3): Send message to -> 4 message (1) local state is 0
1 DEBUG (4): Send message to -> 5 message (0) local state is 1
  DEBUG (5): Send message to -> 6 message (e0) local state is e1
1 DEBUG (6): Send message to -> 7 message (e1) local state is e0
  DEBUG (7): Send message to -> 0 message (1) local state is 0
2 DEBUG (6): Send----> 7      with NOACK
3 DEBUG (6): Send message to -> 7 message (e1) local state is e0
  DEBUG (0): Send----> 1      with NOACK
  DEBUG (0): Send message to -> 1 message (0) local state is 1
2 DEBUG (4): Send----> 5      with ACK
  DEBUG (3): Send----> 4      with NOACK
  DEBUG (3): Send message to -> 4 message (1) local state is 0
  DEBUG (2): Send----> 3      with NOACK
  DEBUG (2): Send message to -> 3 message (e1) local state is e0
3 DEBUG (5): Reply ====>  4  with  NOACK
  DEBUG (5): Send message to -> 4 message (e1) local state is e0
  DEBUG (7): Reply ====>  0  with  NOACK
  DEBUG (7): Send message to -> 0 message (1) local state is 0
  DEBUG (2): Send----> 3      with ACK
  DEBUG (0): Send----> 1      with ACK
4 DEBUG (6): Send----> 7      with ACK
  DEBUG (3): Reply ====>  2  with  ACK
  DEBUG (1): Reply ====>  0  with  ACK
4 DEBUG (5): Reply ====>  4  with  ACK
  DEBUG (7): Reply ====>  6  with  ACK
  DEBUG (1): Send message to -> 2 message (e1) local state is e0
  DEBUG (2): Send message to -> 1 message (1) local state is e1
  DEBUG (1): Send----> 2      with ACK
  DEBUG (2): Reply ====>  1  with  ACK
  DEBUG (3): Send message to -> 4 message (0) local state is 0
  DEBUG (5): Send message to -> 6 message (e1) local state is e0
  DEBUG (6): Send message to -> 5 message (1) local state is e1
  DEBUG (5): Send----> 6      with ACK
  DEBUG (4): Send message to -> 3 message (e1) local state is 1
```

Fig. 6. Messages exchanged for cycle topology.

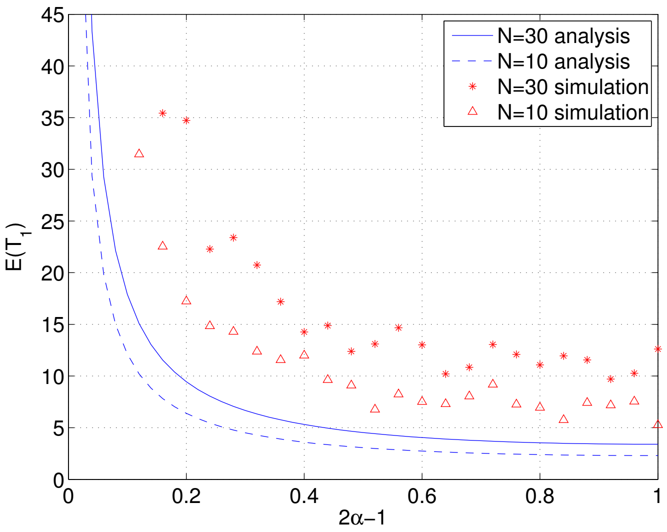


Fig. 7. $E(T_1)$ for star shaped network versus $2\alpha - 1$.

the network. It will be then difficult to converge the system. Otherwise, when α goes to 1, ($2\alpha - 1$ goes to 1), initially, the majority state is held by a number n of nodes close to N, which means that the distributed binary consensus algorithm converges rapidly. In addition, if we consider, as an example, that $\alpha = 0.52$ of the nodes held the majority state, from a network with $N = 10$ nodes and another with $N = 30$ nodes, the average convergence time increases hugely (by a value more than 10 second). However, this increase is limited to 1 second for $2\alpha - 1 = 0.8$. Regarding that the implementation is done in a real life environment, including the routing problems, the ACKnowledgment and NOACKnowledgment mechanism, the average convergence time cannot directly be compared to the analytical results. However, if we consider the evaluation of this time for a network with N nodes, this implies that the ACK and NOACK probably will be the same for two different values of α. The behavior of the average convergence time is still the same for both simulation and analysis. Results presented in Fig. 7 show that the behavior of the average convergence time curve for simulated results is the same as for analytical results. The important random differences can be justified by the ACK and NOACK effects which are a complex phenomenon due to the random reproduction of channel representing the link from node i to node j.

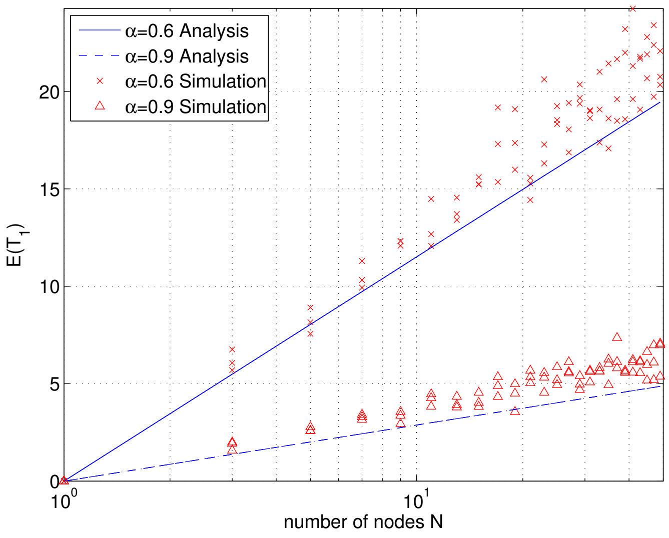


Fig. 8. $E(T_1)$ for a network with complete graph.

The results in Fig 8 show the average convergence time versus the number of nodes in the network for a fully connected topology. For both curves, $\alpha = 0.6$ and $\alpha = 0.9$, the consensus takes more time for a network with a large number of nodes, the number of messages depends on the number of edges which is close to $\frac{N(N-1)}{2}$ (complete graph). In addition, it is shown that the convergence speed depends on α. For a network with $N = 10$ nodes, to reach the consensus values, the system spends 8.55 as an average time from a scenario of $\alpha = 0.9$ to $\alpha = 0.6$. The simulation using TOSSIM characterizes the near real life environment by including the channel path loss, packet loss and ACK/NOACK mechanism. Therefore, the upper bound of the average convergence time, given by analytical results, is not crossed by the simulation points.

In Figs. 9 and 10 the simulation results and the analytical results characterize the behavior of the average convergence time versus the number of nodes. It is noticed that the curves are similar except that the slope of the figure is different by a fraction close to $1/4$ and the initial value of the convergence time starts at $N = 3$ for paths graph with $\alpha = 3/4$. However this same initial value starts at $N = 5$ for ring path with $\alpha = 3/4$. The point representing the simulation value are presented by star (*) and triangle ($^\triangle$) markers. These points are over the average convergence time bound. Again, this is due to real life phenomenon that appear in simulation using TOSSIM such as lost packet due to the link quality or to the congestions. These effects are present by the ACK and NOACK for messages

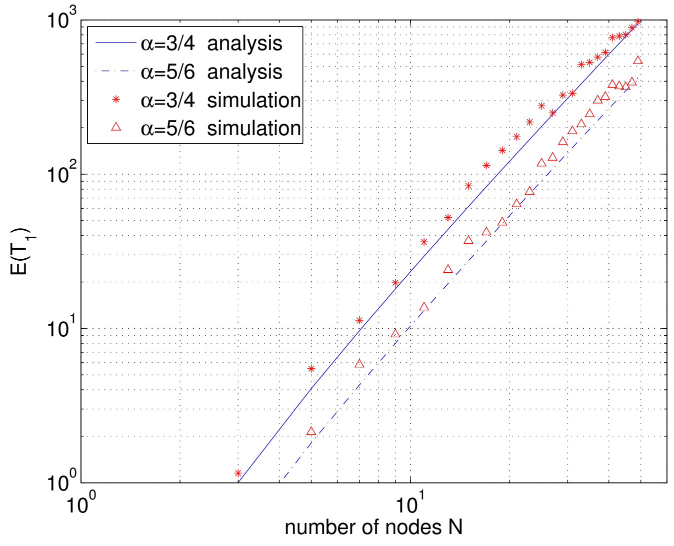


Fig. 9. $E(T_1)$ for ring topology.

traces present by Figs. 5 and 6.

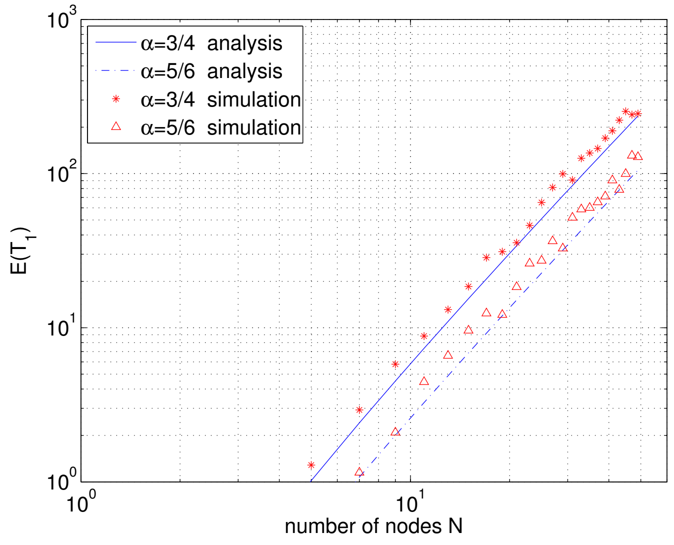


Fig. 10. $E(T_1)$ for cycles network.

VI. Conclusion

In this paper code implementation and evaluation of binary consensus algorithm in WSN is presented for TinyOS and network community. The output in this paper completes the theoretical results presented previously. The evaluation of the algorithm was done on TOSSIM simulator to support the future hardware implementation using a testbed based on IRIS sensor motes from Crossbow.

Some limitations have been noticed during this work. For TinyOS simulation, the same code runs on each sensor node. In addition, some physical phenomenon, such as power consumption, or analog sensed values cannot be easily evaluated. The use of python as the main simulator of WSN, in cooperation with nesc limits the running time and the debugging features of the simulation. Therefore, it will be so important to use c++ as main simulator instead of python to have a real time simulation with complete features.

Acknowledgment

This work is supported by Qatar National Research Foundation QNRF through grant NPRP 09-1150-2-448.

References

[1] M. Draief and M. Vojnovic, “Convergence speed of binary interval consensus,” in *In Proc IEEE Communications Society IEEE INFOCOM 2010*, San Diego CA, USA, March 15-19, 2010.

[2] A. Bashir, A. Akbar, S. Chaudhary, C. Hussain, and K. Kim, “Collaborative detection and agreement protocol for routing malfunctioning in wireless sensor networks,” in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, vol. 1. IEEE, 2006, pp. 327–332.

[3] K. Avrachenkov, M. El Chamie, and G. Neglia, “A local average consensus algorithm for wireless sensor networks,” in *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*. IEEE, 2011, pp. 1–6.

[4] L. Chen, G. Carpenter, S. Greenberg, J. Frolik, and X. Wang, “An implementation of decentralized consensus building in sensor networks,” *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 667–675, 2011.

[5] B. Parhami, “Voting algorithms,” *Reliability, IEEE Transactions on*, vol. 43, no. 4, pp. 617–629, 1994.

[6] P. Bliman and G. Ferrari-Trecate, “Average consensus problems in networks of agents with delayed communications,” in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC ’05. 44th IEEE Conference on*, 2005, pp. 7066–7071.

[7] Y. G. Sun, L. Wang, and G. Xie, “Average consensus in networks of dynamic agents with switching topologies and multiple time-varying delays,” *Systems & Control Letters*, vol. 57, no. 2, pp. 175 – 183, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167691107001120

[8] P. Ezhilchelvan, A. Mostefaoui, and M. Raynal, “Randomized multivalued consensus,” in *Object-Oriented Real-Time Distributed Computing, 2001. ISORC - 2001. Proceedings. Fourth IEEE International Symposium on*, 2001, pp. 195–200.

[9] A. Mostefaoui, M. Raynal, and F. Tronel, “From binary consensus to multivalued consensus in asynchronous message-passing systems.” *Information Processing Letters*, no. 73(5-6), pp. 207–212, 2000.

[10] E. Perron, D. Vasudevan, and M. Vojnovic, “Using three states for binary consensus on complete graphs,” in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2527–2535.

[11] F. Benezit, P. Thiran, and M. Vetterli, “Interval consensus: from quantized Gossip to voting,” in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 3661–3664.

[12] K. Alekeish and P. Ezhilchelvan, “Consensus in sparse, mobile ad-hoc networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 3, pp. 467 – 474, march 2012.

[13] Y. Mostofi and Y. Yuan, “Impact of heterogeneous link qualities and network connectivity on binary consensus,” in *American Control Conference, 2009. ACC’09*. IEEE, 2009, pp. 1821–1826.

[14] Y. Wang and P. Djuric, “Reaching consensus on a binary state by exchanging binary actions,” in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3297–3300.

[15] Y. Mostofi and M. Malmirchegini, “Binary consensus over fading channels,” *Signal Processing, IEEE Transactions on*, vol. 58, no. 12, pp. 6340–6354, 2010.

[16] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” *Information Theory, IEEE Transactions on*, vol. 52, no. 6, pp. 2508–2530, 2006.

[17] Y. Ruan and Y. Mostofi, “Binary consensus with soft information processing in cooperative networks,” in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*. IEEE, 2008, pp. 3613–3619.

[18] L. Zheng and Y. Yao, "Binary decision consensus in ad hoc sensor network," in *Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th*. IEEE, 2009, pp. 1–5.

[19] Y. Lan and S. Huang, "Designing an efficient collaborative learning model to construct a consensus based on binary tree structure," in *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*. IEEE, 2009, pp. 182–187.

[20] S. Shang, P. Cuff, S. Kulkarni, and P. Hui, "An upper bound on the convergence time for distributed binary consensus," in *Information Fusion (FUSION), 2012 15th International Conference on*. IEEE, 2012, pp. 369–375.

[21] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137.

[22] M. Demmer and P. Levis, "Tython scripting for TOSSIM," *Network Embedded Systems Technology Winter*, 2004.

[23] S. Notani, "Performance simulation of multihop routing algorithms for ad-hoc wireless sensor networks using tossim," in *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, vol. 1. IEEE, 2008, pp. 508–513.

[24] M. Safaei and A. Ismail, "SmartSim: Graphical Sensor Network Simulation Based on TinyOS and TOSSIM," in *Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on*. IEEE, 2012, pp. 611–615.

[25] P. Buonadonna, J. Hill, and D. Culler., "Active message communication for tiny networked sensors," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM'01)*, April 2001.

[26] A. Woo and D. E. Culler, *Evaluation of efficient link reliability estimators for low-power wireless networks*. Computer Science Division, University of California, 2003.

[27] M. Asikainen, K. Haataja, R. Honkanen, and P. Toivanen, "Designing and Simulating a Sensor Network of a Virtual Intelligent Home Using TOSSIM Simulator," in *Wireless and Mobile Communications, 2009. ICWMC'09. Fifth International Conference on*. IEEE, 2009, pp. 58–63.

[28] R. Gao, H. Zhou, and G. Su, "A wireless sensor network environment monitoring system based on tinyos," in *Proceedings International Conference on Electronics and Optoelectronics (ICEOE'11)*, vol. 1, Beijing CHINA, 29-30 Jully 2011, pp. 497–501.

[29] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM Sigplan Notices*, vol. 35, no. 11, pp. 93–104, 2000.

[30] C. Technology. Professional kit for wireless sensor networks. [Online]. Available: http://www.xbow.com

[31] C. T. Inc. IRIS wireless measurement system datasheet. [Online]. Available: http://www.xbow.com/Products/Productpdffiles/Wirelesspdf/IRISDatasheet.pdf

[32] P. Levis. Tinyos programming. [Online]. Available: http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf