Computer Networks 79 (2015) 30–38

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Distributed algorithms in wireless sensor networks: An approach for applying binary consensus in a real testbed

Noor Al-Nakhala *, Ryan Riley, Tarek Elfouly

Qatar University, Department of Computer Science and Engineering, Doha, Qatar

ARTICLE INFO

Article history:
Received 6 February 2014
Received in revised form 15 October 2014
Accepted 17 December 2014
Available online 7 January 2015

Keywords:
Binary consensus
TinyOS
Wireless sensor networks

ABSTRACT

In this work, we realize the binary consensus algorithm for use in wireless sensor networks. Binary consensus is used to allow a collection of distributed entities to reach consensus regarding the answer to a binary question and the final decision is based on the majority opinion. Binary consensus can play a basic role in increasing the accuracy of detecting event occurrence. Existing work on the binary consensus algorithm focuses on simulation of the algorithm in a purely theoretical sense. We fill the gap between the theoretical work and real hardware implementation by modifying the algorithm to function in wireless sensor networks. This is achieved by adding a method for nodes to determine who to communicate with as well as adding a heuristic for nodes to know when the algorithm has completed. Our implementation is asynchronous and based on random communication. In this work, we expand our previous implementation to test it on 139 hardware testbed. Moreover, we are able to minimize the convergence time achieving ultimate results. Our implementation show successful results and all the motes are able to converge to the expected value in very short time.

1. Introduction

Algorithms for cooperative decision making have received significant attention in recent years. In these algorithms, a network of agents seeks to reach a decision cooperatively and ensure that all nodes in the network know the final decision. The consensus problem comes when the agents should agree on a certain value. One such algorithm in this area is binary consensus [1,2]. Under binary consensus, the nodes in the network must simply agree on whether a statement is TRUE or FALSE. For example, a network of nodes capable of measuring temperature could use binary consensus to answer the question "Is the temperature greater than 80 °C?" in order to help detect a fire in a building.

In the binary consensus problem, each node has an initial state of either 0 or 1, and the nodes should decide which one of these values are correctly held by the majority of the nodes in the network.

The existing algorithm for binary consensus has two limitations. First, it does not specify how nodes find partners to run the algorithm with. Second, it does not provide a way for an individual node to determine when consensus has been reached. In order to implement the algorithm in a real distributed network, these limitations must be overcome.

Wireless sensor networks consisting of small, embedded devices, called motes, provide an excellent platform for binary consensus. Motes contain sensors that can be used to collect data about their environments, and can communicate with each other wirelessly [3]. Due to limitations regarding their size and power, sensor motes are computationally weak and should limit the number of packets they send. When data is transmitted by the sensor

* Corresponding author.
E-mail addresses: nalnakhala@qu.edu.qa (N. Al-Nakhala), ryan.riley@qu.edu.qa (R. Riley), tarekfouly@qu.edu.qa (T. Elfouly).

http://dx.doi.org/10.1016/j.comnet.2014.12.011
1389-1286/

motes in the network, more energy is consumed in the process of transmission than the process of computation [4].

Previous research focused only in the theoretical side of the binary consensus without going further to implement this algorithm in real sensor motes. The binary consensus algorithm can be effectively used in wireless sensor motes in numerous fields to increase the accuracy of detecting certain decisions or events.

To the best of our knowledge, there is no exiting work that deals with implementing binary consensus algorithm in real world scenario. Our goal in this work is to study binary consensus algorithm further by implementing it in real world implementation.

In [5] we implemented and tested our algorithm in real wireless sensor motes by applying it in 11 sensor motes, and further support our results with more motes and topologies in a wireless mote simulator. Our previous results of convergence time showed that convergence speed depends on the following factors: the topology, the number of motes present in the network and the distribution of the initial 0 and 1 states. In this paper, we propose a set of modifications to binary consensus that will allow it to operate in the context of wireless sensor motes having the limitations described above. Our modifications consist of changing how motes decide who to communicate with and also adding a heuristic to help motes estimate when consensus has been achieved. We have implemented our algorithm in a set of TinyOS based sensor motes and verified our algorithm functions both in hardware and in simulation. Moreover, we tested our implementation in a large sensor network consists of 139 motes.

2. Background

In this section we will give a brief overview of binary consensus and potential applications of it to wireless sensor networks (WSNs). We assume the reader is familiar with WSNs, and focus on binary consensus here.

2.1. Binary consensus

There are a variety of algorithms that are meant to allow a network of distributed nodes to reach consensus in a computation. In this work, we are specifically concerned with the problem of *binary consensus* [6,7], where each node in the network holds one of two states and the algorithm allows all nodes to learn which state is held by the majority of nodes. There are many applications of such an algorithm, such as determining if the majority of sensors in a network have observed a certain event. Two strengths of binary consensus are that it is guaranteed to come the correct conclusion [7], and that there is an upper-bound on the time to convergence [1].

Under binary consensus, nodes in the network start with their initial state and then update their state with each other based on an updating protocol. Convergence occurs when all nodes agree on the majority opinion. When two nodes communicate and run the updating protocol, they compare current states and then each assume a new state based on what they have seen. While the algorithm is running a node may be in one of four states, which can be described informally as:

1. 0 – The node believes the majority opinion is most likely false.
2. e_0 – The node believes the majority opinion might be false.
3. e_1 – The node believes the majority opinion might be true.
4. 1 – The node believes the majority opinion is most likely true.

The updating protocol, as quoted from [1], is as follows:

Each node is in one of four states: 0, e_0, e_1, and 1. The states satisfy the following order $0 < e_0 < e_1 < 1$. At each contact of a pair of nodes, their respective states x and y (without loss of generality) ordered such that $x \leqslant y$, are updated according to the following mapping $(x, y) \mapsto (x', y')$ defined by

$$\begin{aligned}
(0, e_0) &\rightarrow (e_0, 0)\\
(0, e_1) &\rightarrow (e_0, 0)\\
(0, 1) &\rightarrow (e_1, e_0)\\
(e_0, e_1) &\rightarrow (e_1, e_0)\\
(e_0, 1) &\rightarrow (1, e_1)\\
(e_1, 1) &\rightarrow (1, e_1)\\
(s, s) &\rightarrow (s, s), \text{ for } s = 0, e_0, e_1, 1.
\end{aligned}$$

Convergence occurs when all nodes have states $\in \{0, e_0\}$ or $\in \{e_1, 1\}$. This means that if all nodes in the network have state 0 or e_0, then the network has converged and the majority of nodes initially held the value 0. Likewise, if all nodes in the network have state e_1 or 1, then the network has converged and the majority of nodes initially held the value 1.

Consider the following theoretical example of how the binary consensus algorithm works, independent of the implementation methodology. Assume that there is a network with 4 nodes, 1, 2, 3 and 4 having initial states of $(1; 0; 0; 0)$ respectively as shown in Fig. 1(a). The first interaction happens between nodes 1 and 2 and the state of node 1 becomes e_0 while the state of node 2 will be e_1. (This is according to the rules given above.) So the new sequence of states will be $(e_0; e_1; 0; 0)$. Next, the second interaction is between nodes 3 and 4 as shown in Fig. 1(b); they communicate and nothing happens since they both hold the same state. Now, nodes 1 and 2 communicate again as depicted in Fig. 1(c), and their states are swapped leading to $(e_1; e_0; 0; 0)$. Nodes 2 and 3 communicate as illustrated in Fig. 1(d) and also swap their states: $(e_1; 0; e_0; 0)$. Finally, node 1 communicates with node 2 as shown in Fig. 1(e) leading to the converged states $(0; e0; e0; 0)$ illustrated in Fig. 1(f). We consider this set of states converged because all nodes have value 0 or e_0. This means that the majority of nodes initially held state 0.

It is important to note that even though convergence has occurred, the nodes continue to communicate and exchange states. This is because individual nodes do not