Efficient Distributed Average Consensus in Wireless Sensor Networks

Christophe Guyeux[1], Mohammed Haddad[2], Mourad Hakem[1] and Matthieu Lagacherie[1]
[1]Disc Laboratory, Femto-ST Institute, UMR 6174 CNRS, Université de Bourgogne Franche-Comté, France
[2]LIRIS Laboratory, UMR CNRS 5205, Université de Lyon 1, F-69622, France
Authors are in alphabetic order

***Abstract*— Computing the distributed average consensus in Wireless Sensor Networks (WSNs) is investigated in this article. This problem, which is both natural and important, plays a significant role in various application fields such as mobile agents and fleet vehicle coordination, network synchronization, distributed voting and decision, load balancing of divisible loads in distributed computing network systems, and so on. By and large, the average consensus' objective is to have all nodes in the network converged to the average value of the initial nodes' measurements based only on local nodes' information states. In this paper, we introduce a fully distributed algorithm to average the sensed data within the network itself. The network may be large since we never broadcast over all its nodes. Unlike earlier works, when a node detects a load (scalar value) imbalance in its closed neighborhoods during the average process, instead of sending parsimonious amount of load values from highly loaded nodes to less loaded ones, we move a large amount of load values by involving parallel atomic transactions between mutually exclusive pairs of neighbors. This improves the global convergence time speedup with low-cost communication and minimal energy consumption. First, we give the convergence proof of the distributed consensus process, and next we provide some experimental results based on NS3 framework to assess the behaviour of the proposed algorithm.**

I. Introduction

With the technological progress of WSN and IoT, the design of efficient protocols for in-network average consensus has received a great interest and attention. It is motivated by diverse potential needs: resiliency, throughput, resources' usage, and rapid response time. They constitute naturally reliable and efficient algorithms for large scale distributed information processing, and do not necessitate any information knowledge of the network . Indeed, centralized sensor networks have limited scalability and are less robust to node failures, since a full network's reorganization should take place whenever a node fails or had just been added to the network. The objective of average consensus is to orchestrate the propagation of the initial node's measurements, so that the difference between the converged nodes' values in the network is reduced as low as possible with low time complexity. This problem arises in different fields of applications. Some common practical examples are outlined hereafter:

– *Voting* in distributed networks: every node initially produces a binary bit of information: 0 or 1. The voting process consists in letting all nodes know whether 0 or 1 was the bit with higher density (the initial majority bit), based only on local interactions [6].

– *Unattended Wireless Sensor Networks (UWSNs)*: these networks are widely used in diverse fields of applications like pipeline monitoring and oceanography. The objective is to ensure data survivability despite the hostile conditions of the environment and the sporadic presence of the sink. Indeed, the sink's absence for a period of time will prevent sensors to upload information in real time. One possible solution to this problem is to average the collected data and then upload the obtained information once the sink becomes available [3].

– *Coordination* of a fleet of vehicles: local values indicate the velocity or the position of each vehicle, and the distributed consensus is to obtain the best coordination in steady-state that gives the smallest mean-square synchronization error [8].

– *Load balancing* of divisible loads in distributed computing networks: the objective is to distribute fairly the global workload to minimize as low as possible the load difference between the networks' resources. In divisible loads model, the parallel application includes no precedence constraints and granularity of parallelism is fine. One example among many is an application of records search in a huge databases that may be divided into independent parts of one record granularity [12]. This can be done by cooperating the system's resources and the search can be performed in each part independently. The results are finally sent to some centralized master node.

– *Distributed agreement and decision*: a whole network needs to agree on a suitable decision to take in order to perform some specific coordinated operations [6]. For example, a fleet of robots needs to move in the same direction, or a WSN whose purpose is to track enemies wants to agree on the presence of such an enemy.

– *Clock synchronization* in distributed systems: sensors should agree on a common time and synchronize their clocks. Local values can represent the reading of a local clock values corrupted by random noise and adjusted via a diffusion scheme [28], [29], [24]. The objective is to choose the best edge weights that minimize the mean square error.

In this paper, we present a fully distributed algorithm for the problem of average consensus in wireless sensor networks based on *mutually exclusive atomic transactions* between neighbors. The network may be arbitrary and large since we never broadcast over its sensor nodes. Unlike earlier works, upon load (scalar value) imbalance detection between neighbors, instead of sending parsimonious amount of load values from highly loaded nodes to less loaded ones, we move a large amount of load values by involving *parallel mutually exclusive transactions* between pairs of

neighbors. That is to say, each involved pair-neighbors seek to update their estimate values to both take on the atomic mean of their previous estimates. Based on NS3 simulator, several scenarios are studied and different metrics are evaluated. Obtained results show that the proposed algorithm achieves good performances compared to its direct competitor, namely the well-known Bertsekas and Tsitsiklis' algorithm [5], in terms of communication costs, energy consumption, and convergence time speed. The presented work targets WSN platforms with the aim of computing the average of initial sensors' measurements within the network. It is easily extendable to the case of load balancing of divisible loads in distributed computing networks. To achieve global stable equilibrium, processor nodes exchange load with their neighbors iteratively and update their local load until reaching the final balance load distribution. Note that, throughout the paper, we use the terms "load" and "scalar" indifferently.

The remainder of this article is organized as follows. Section II is devoted to the review of related work. Since we compare our proposal to Bertsekas and Tsitsiklis' algorithm, we outline its principle in Section III. Section IV presents our proposed algorithm. We report in Section VI series of numerical results that assess the good behavior of our algorithm. Finally, Section VII summarizes the results of this work and concludes the paper.

II. Literature review

Several techniques have been proposed in the literature to tackle the problem of average consensus in distributed networks systems. In [29], [30], for instance, a decentralized fusion scheme based on average consensus is investigated. Instead of involving point to point communication or any rooting protocol, the information is diffused in the network by bringing up-to-date each node's data value with a weighted average of its neighborhood. The global convergence is guaranteed even with unreliable communication links. In [18], a consensus propagation which can be seen as a special case of belief propagation [25] is proposed to average number values in distributed networks systems. The authors show its convergence property for regular network graphs and its scalability with network's density.

In [11], both consensus and gossip algorithms for ergodic networks are addressed. Based on Oseledets theorem, some mathematical tools are derived to evaluate and characterize several metrics: convergence cost, message passing, and energy efficiency of the studied algorithms. To coordinate a fleet of unmanned vehicles, a distributed consensus-based bidding is investigated in [8]. The proposed approach is based on both consensus and bidding to converge a fleet of vehicles to a consistent situational awareness and conflict-free assignment over different network topologies. In [14], [15], [16], [23], the authors propose a fault tolerant communication scheme for average consensus in networks subject to link failures. Based on Markov processes and stochastic approximation theory, they derive a trade-off relation between the mean-squared error and the algorithm's convergence time.

Some approaches like [20], [19], [21], [22] are based on mobile agents or Kalman filtering to calculate the global average consensus of the initial node's measurements in the network. Other techniques [17], [31], [27] make use of linear iterations, where each node in the network updates its local value by a weighted sum of its neighborhood's data. This leads to efficient convergence time and error variance reduction over the network.

In [26], the authors use a concept of domination in graphs to average the initial nodes' measurements. More precisely, the information exchange between nodes involves only the connected dominating set (CDS) of the network graph to reduce the sensors' overall power consumption. [4] developed a novel approach using self stabilization under a serial daemon scheduler. It was shown that the proposed algorithm converges in a linear time complexity with a tighter bound of the global equilibrium threshold. The authors in [32] study clustering approaches for average consensus based on gossiping in hierarchical WSNs. The aim of the presented work is to organize the targeted network into clusters. Each cluster head computes its local average consensus and then broadcasts the obtained value to all nodes in the cluster. Clusters are chosen randomly, and the wake up scheme of the cluster heads use a Poisson process as a time model to improve the convergence time of the proposed algorithm.

Average consensus can also be seen as a special case of divisible load balancing problem in distributed computing networks. Such computational loads can be divided into parts of small sizes that can be executed independently in parallel by different nodes in the system. The first landmark work in this context is proposed in [5]. The authors assume that the computational load consists of the execution of independent divisible tasks and load transfer between processors takes place asynchronously. The algorithm is guaranteed to converge to the final balanced state under some conditions and hypothesis (see Section III for more details). This work has been adapted in other works. For instance, in [2], [1], the authors deal with dynamic networks where communication links between the resources of the network are intermittent. In [9], the proposed work investigates the problem of static divisible integer loads in heterogeneous networks. The algorithm is iterated until the load difference between any pair of neighbor nodes is one load unit. The same work is extended in [7] to consider partially asynchronous discrete load model in parallel computers networks for which the convergence proof is also provided.

A fast local load balancing approach for ring networks is presented in [13]. The addressed work considers both synchronous and asynchronous models and assumes static load situations where no tokens are generated during the load balancing process. The authors show that the algorithm converges to the balanced distribution state with low time complexity. A diffusion approach for load balancing in hypercube networks is addressed by Cybenko in [10]. The author considers both static and dynamic workload

distribution and assumes equal tasks' amount of computation times. Based on the network's iterative diffusion matrix, the convergence properties to the uniform load distribution are derived.

III. A Brief description of Bertsekas and Tsitsiklis' Algorithm

In order to compare our proposal to the well-known Bertsekas and Tsitsiklis' algorithm, which is, to our knowledge, the closest work to the one presented in this work, we briefly outline hereafter the key features of this algorithm.

Consider a system network of n nodes, represented by a connected undirected graph $G = (N, A)$, where $N = \{1, \ldots, n\}$, and A is the set of links between nodes. Let $A(i)$ be the set of neighbors of node i and $x_i(t)$ be the node's local load at time t. The key idea of Bertsekas and Tsitsiklis' algorithm is that each node cooperates with its neighbors in asynchronous diffusion mode to reach the global equilibrium. At each time step t, each node i gets the state of its neighbor j at time $d^i_j(t)$, where $0 \leq d^i_j(t) \leq t$. Thus, $d^i_j(t)$ stands for the communication delay between i and j, and $x^i_j(t) = x_j(d^i_j(t))$ denotes the jth node's local state at time $d^i_j(t)$, received by node i at time t.

The value $s_{ij}(t)$ represents the quantity of load sent by i to j at time t, and $r_{ji}(t) = s_{ji}((d^i_j(t))$ is the amount of load received by i from j at time t. Each node i keeps an estimate $x^j_i(t)$ of the load carried by each neighbor j. Then, the updated load value of node i at time $t+1$ is given by the following formula:

$$x_i(t+1) = x_i(t) - \sum_{j \in A(i)} s_{ij}(t) + \sum_{j \in A(i)} r_{ji}(t)$$

It is assumed that each node i is connected to a node j within any asynchronism time interval measure of length B, and this delay cannot greater than B. Formally:

There exists $B \in \mathbb{N}$ such that $\forall t \geqslant 0,\ t - B < d^i_j(t) \leq t$ and the union of communication graphs $\bigcup_{\tau=t}^{t+B-1} G(\tau)$ is a connected graph.

The asymptotic convergence is based upon the following assumptions:

Assumption 1: There exists some constant $\alpha > 0, \forall t \geqslant 0$,
$\forall i \in N, \forall j \in A(i)$, such that $s_{ij}(t) \geq \alpha(x_i(t) - x^i_j(t))$.
Moreover, if $(x_i(t) \leq x^i_j(t))$, then $s_{ij}(t) = 0$.

Assumption 2: $x_i(t) - \sum_{k \in A(i)} s_{ik}(t) \geq x^i_j(t) + s_{ij}(t)$.

The first condition assumes that, when a given node detects a load imbalance with its neighbors, it will send some amount of its excess load to its less loaded neighbors. The second one prohibits the ping-pong phenomenon where two nodes keep transferring parts of their own load to each other without reaching equilibrium.

IV. The Proposed Algorithm

The proposed algorithm takes its inspiration from the communicating vessels phenomena. A vessel that is less loaded than a connected neighbor will create a depression that will cause the more loaded vessel to transfer some of its load until equilibrium is reached. Given a graph $G = (V, E)$ modelling the network, transfers are defined similarly between high loaded nodes and lesser loaded ones. The algorithm should converge until no more load transfer occurs.

Algorithm 1: Distributed Consensus Algorithm (DCA)

Nodes: i is the current node, j is a neighbor of i
Constants: ε is the local equilibrium threshold
Variables: $i.x$ is the current value, $i.I$ is the Invitation pointer, $i.A$ is the Acceptance pointer
Macros: $N(i)$ is the set of neighbors of i

Pointer Correction Rules:

IF $i.I = j \wedge (j \notin N(i) \vee j.x - i.x \leq \varepsilon \vee j.I \neq null)$
Then $i.I := null$; [C1]

IF $i.A = j \wedge (j \notin N(i) \vee j.I \neq i \vee i.x - j.x \leq \varepsilon)$
Then $i.A := null$; [C2]

Transfer Transaction Rules:

IF $i.I = null \wedge i.A = null \wedge$
$\exists j \in N(i) : j.x - i.x > \varepsilon \wedge j.I = null$
Then $i.I := j$; [I]

IF $\exists j \in N(i) : i.x - j.x > \varepsilon \wedge$
$j.I = i \wedge i.A = null \wedge i.I = null$
Then $i.A := j \wedge i.x := \dfrac{i.x + j.x}{2}$; [A]

IF $\exists j \in N(i) : j.A = i \wedge i.I = j$
Then $i.x := j.x$; [F]

IF $\exists j \in N(i) : ((j.A = i \wedge i.I = j) \vee$
$(j.I = i \wedge i.A = j)) \wedge i.x = j.x$
Then $i.I := i.A := null$; [R]

The average consensus is reached within an accuracy σ on the global equilibrium threshold. The global legitimate state of the network is then defined as follows:

$$\forall i, j \in V : |x_i - x_j| \leq \sigma \quad (1)$$

Respectively to the global equilibrium threshold σ, we define the local equilibrium threshold ε which has to be chosen such that Expression (1) is respected. A trivial solution is to define the local threshold ε according to the diameter of the network D, by setting $\varepsilon < \frac{\sigma}{D}$.

The algorithm is written according to the composite read-write atomicity model, in which each node i can read its variables and those of its direct neighbors, but can only write on its own variables. The algorithm is then defined as a set of rules, each ensuring nodes to perform correct actions according to different events. All variables, macros, and the set of rules a node may execute are defined in Algorithm 1. A node remains active as long as at least one of its rules is active. The convergence is reached once all nodes have all their rules inactive.

Each node i maintains three variables: $i.x$ that indicates the current value at node i, $i.I$ an invitation pointer, and $i.A$ an acceptation pointer. A node i having a neighbor j with a value $j.x$ higher than $i.x + \varepsilon$ invites it to perform a transfer transaction to ensure a pairwise local equilibrium. However i invites j only if the latter is not inviting someone else simultaneously. This is done by the rule `[I]`, that is, the invitation rule. If invited by a neighbor j, a node i checks if all its pointers are free as well as the condition on the gap between respective x values is respected, then it accepts invitation by setting its pointer $i.A$ to j and begins transfer transaction by setting its variable $i.x$ to $\frac{i.x + j.x}{2}$. Acceptation rule is named `[A]` in the algorithm. Thus, when a node i observes that its invitation is accepted, it executes the transaction finalization rule `[F]` and updates its $i.x$ variable consequently. Once the transaction finalized, both neighbors will execute the pointers reset rule `[R]`. This mechanism is executed as long as there is neighbors that are not in local equilibrium state. Moreover, note that transfer transaction rules are only executed if there is no incoherence in pointers $i.I$ and $i.A$. Rules `[C1]` and `[C2]` verify the coherence of invitation and acceptation pointers respectively. These two correction rules are checked prior to any transaction rule.

V. Algorithm's analysis

In this section, we give the convergence proof of the proposed algorithm, that is, the number of the atomic transfer transactions towards the global equilibrium is finite.

Let $\alpha(t)$ and $\beta(t)$ be the maximum value, resp. the minimum value in the sensor network at the time t. Let ε be the local equilibrium threshold.

Lemma 1: The set of transfer transactions T between sensor nodes are independent: $T = \{ij \mid i.I = j \wedge j.A = i\}$.

Proof: According to the the invitation rule `[I]`, a sensor node can make/have at most one invitation/preference at a time. Thus, there can be never two concomitant transfer transactions in T. ∎

Lemma 2: $\alpha(t)$ (rsp. $\beta(t)$) is monotonically decreasing (rsp. increasing).

Proof: When we deal with distributed average consensus (or load balancing in distributed systems), we may have situations where node's local equilibrium is jeopardized by concomitant neighbors' load transfers. In order to avoid this effect, loads are moved from highly loaded sensors to less loaded ones in atomic transactions. In addition, as pointed by Lemma 1, the set of transfer transactions nodes are not concomitant. Hence, $\forall t, \alpha(t) \geq \alpha(t+1)$ and $\beta(t) \leq \beta(t+1)$. ∎

Lemma 3: The invitation process is not chaining.

A node i is chaining means i has selected j but j has invited another node.

Proof: By contradiction, assume that the invitation pointers are chaining. Observe that such behavior is impossible by applying rule `[I]`. However, the pointers may be affected and vary due to unexpected perturbations. In this case, a pointer correction rule `[C1]` is applied. ∎

Lemma 4: If the network is not balanced, then

$$\alpha(t + \Delta t) < \alpha(t)$$

where Δt is within $\mathcal{O}(n)$ transfer transactions.

Proof: We consider the worst case where all nodes in the network are holding the maximum value except for one node which is not in equilibrium state. In this configuration, the last amount of load in transit before reaching equilibrium could circulate on an Hamiltonian path before arriving to the final node not holding the maximum value. It follows that α (rsp. β) will be decreased (rsp. increased) by at least ε in at most $\mathcal{O}(n)$ transfer transactions. ∎

Theorem 1: The proposed algorithm converges to the final stable equilibrium within $\mathcal{O}\Big(n \times (\alpha(0) - \beta(0))/\varepsilon\Big)$ transfer transactions.

Proof: We have seen by previous Lemmas that α (rsp. β) value is decreased (rsp. increased) by at least ε within $\mathcal{O}(n)$ transactions. The worst case scenario may occur when the consensus value is close to either $\alpha(0)$ or $\beta(0)$. This leads to $\mathcal{O}\Big((\alpha(0) - \beta(0))/\varepsilon\Big)$ atomic transfer transactions before reaching consensus. It follows that the algorithm's convergence cost is at most $\mathcal{O}\Big(n \times (\alpha(0) - \beta(0))/\varepsilon\Big)$. ∎

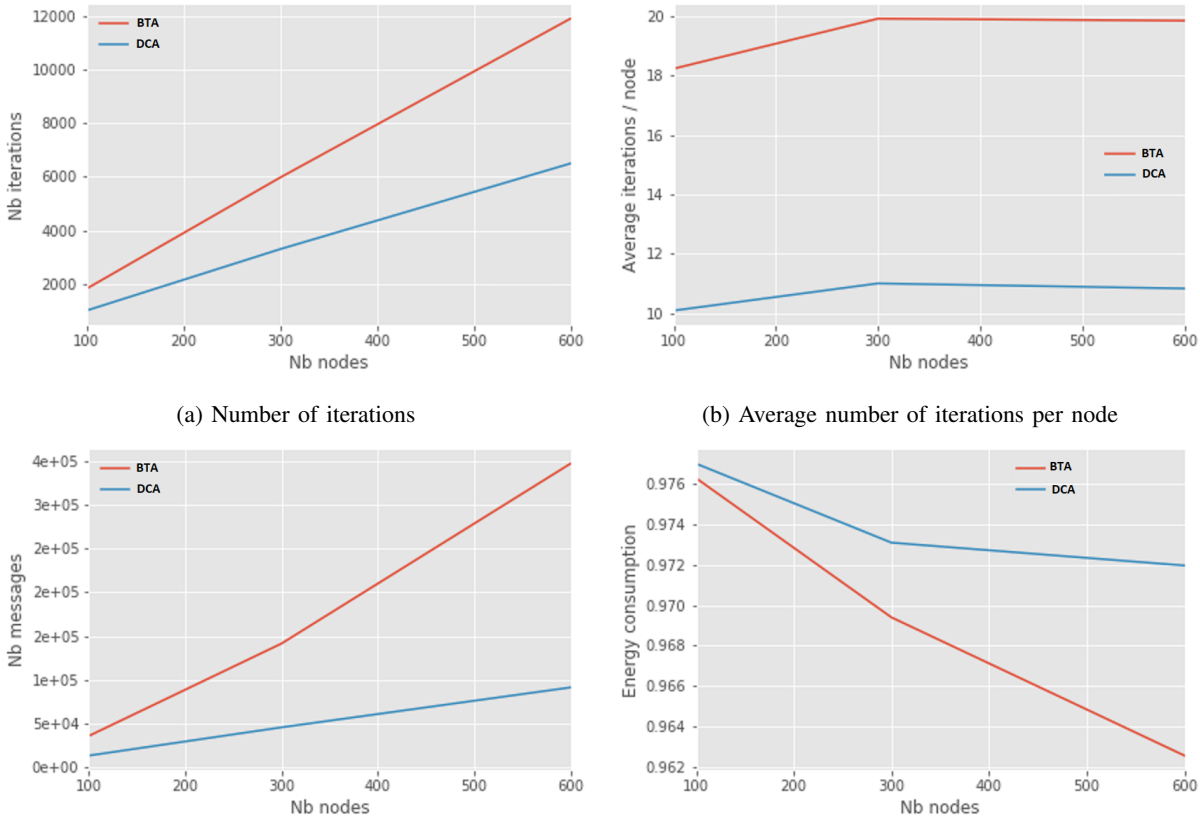
VI. Experimental Protocol and Results

In our experimentation, we compared our algorithm with algorithm presented in Section III. This section describes the steps involved in our experimental run from graph creation to final results analysis. To evaluate the algorithms, we conducted a series of experiments using the network simulator NS3[1]. NS3 is an open-source discrete event simulator primarily used for research and teaching. NS3 is aligned with state of the art networking components and is built using C++ and Python. NS3 modules can be scripted efficiently with Python.

For each experiment, a graph is generated randomly on a simulated map of $[0 - x] \times [0 - x]$ meters, where x depends on the node density (default value of 100 nodes per km^2). In this step, uniform distribution has been used to generate the positions of the nodes. Two nodes are connected if the Euclidean distance between them is less than D meters (simulating a signal range of D meters). A post-processing step is applied to the generated graph to ensure that the graph is fully connected. This step iterates on all the nodes and extracts the connected components. If the number of connected components is greater than 1, then a connection is formed to connect the graph components between them.

In our setup, the sensor networks are modeled by random graphs, edges representing links between nodes. We considered different numbers of nodes n for the sensor networks: 50, 100, 200, 400, and 600. The random graphs are generated by a uniform distribution in a flat-grid with

[1]NS-3: Network simulator 3. Presentation by Gustavo Carneiro at UTM Lab Meeting, April 2010.



(a) Number of iterations

(b) Average number of iterations per node

(c) Number of messages exchanged

(d) Energy consumption

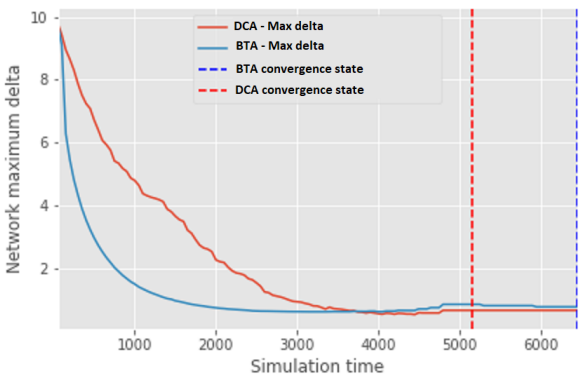
Fig. 1: Iterations, Messages, and Energy consumption

an average density of 100 nodes per km^2. Thus, two nodes are connected if their distance is less than 300 meters. The initial value x of a sensor is a random value following a uniform distribution in the range $[0 - 10]$. The threshold ε, defined in Section IV is set to 0.1. For each size of graph n, we consider 40 executions of the algorithms then the results are averaged. To evaluate the energy consumption of the compared algorithms, a Wifi Radio Energy Model is used. Four states are defined in NS3: TX (Transmission), RX(Transmission), IDLE, and SLEEP with default power consumption values (in Watts) : $TX = 1.14$, $P_{RX} = 0.94$, $P_{IDLE} = 0.82$, and $P_{SLEEP} = 0.10$. Each node has an initial energy of 100 Joules. Note that, in all following figures, Bertsekas and Tsitsiklis' algorithm is denoted by BTA and our algorithm by DCA (Distributed Consensus Algorithm).

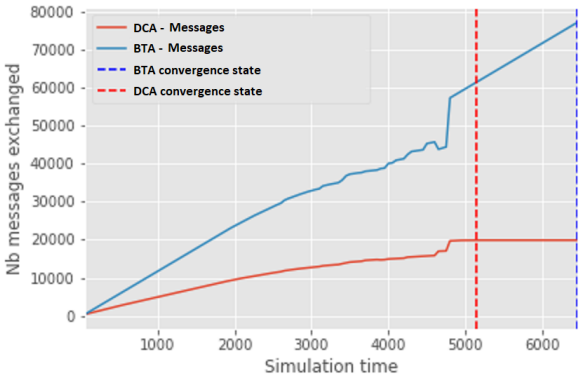
The first bloc of results is given by Figure 1. It shows global and per node iterations numbers, numbers of exchanged messages and energy consumption, all according to number of nodes in the network. We can observe that global iterations numbers of both BTA and DCA algorithms grow linearly in number of nodes. However, numbers of iterations per node stabilize after 300 hundred nodes. Number of exchanged messages is clearly related to number of iterations and we note that DCA algorithm is less verbose than BTA and consumes less energy. The apparent weakness of BTA algorithm is due to the policy adopted for balancing the local load between neighbors. Nodes are chosen so that the ping-pong phenomenon will not occur during the load balancing process. Doing so, the transferred quantities of load are, in fact, not moved to neighbouring nodes that would reach local equilibrium in fewer iteration steps.

Figures 2 and 3 present a finer overview of results on networks of size 100 nodes and 600 nodes respectively. Each of them, in addition to evolution of number exchanged messages in time, shows the evolution of the Delta *i.e.* the difference between the largest and smallest values in the network. Since final Delta values are very similar for both BTA and DCA algorithms, note that evolution of min and max values are given for several runs of DCA algorithm. However, we can observe that final value of Delta is reached more quickly by BTA algorithm than ours, but BTA fails to reach quicker convergence. Indeed, when local unbalanced situations are detected, assigning a part of load in a non suitable way between node neighbors will increase the number of iteration steps before reaching the final balanced load distribution.

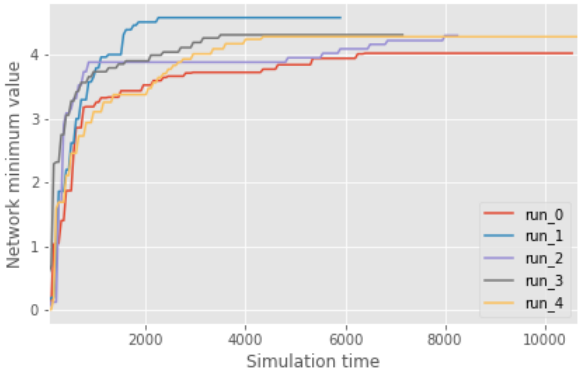
Preserving the sensors' energy is one of the most important issues in designing WSN. Indeed, each sensor node is a small sized device with a limited battery life which, in



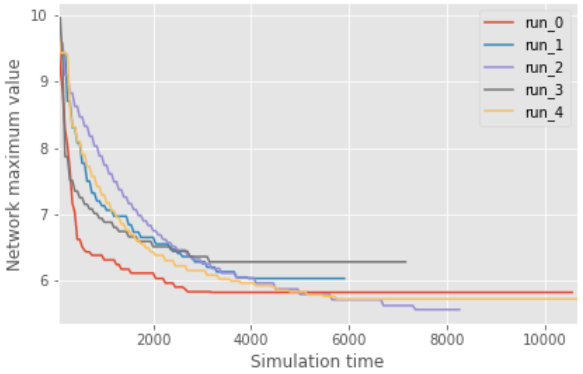
(a) Maximum delta convergence



(b) Number of exchanged messages



(c) Network minimum value



(d) Network maximum value

Fig. 2: Experimentation results with 100 nodes

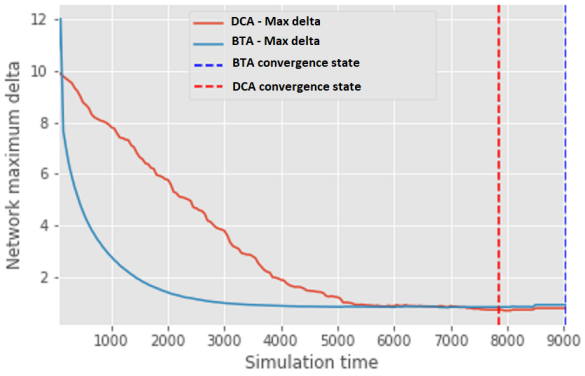
most cases, is non refillable especially in remote and hostile environments. Consequently, in-network average consensus process with minimal energy consumption needs to be considered. In this experiment, to assess the energy saving metric, we plot in Figures 1d and 4 the rate of energy consumption required by the compared algorithms toward the global equilibrium convergence. We can observe that DCA algorithm is more energy-efficient than BTA. Our proposal tends to exhibit a uniform energy consumption and is in line with network's size. However, the inherent nature of BTA algorithm requires higher number of iteration steps and exchanges more messages than DCA. The nodes energy are depleted at a much faster rate leading to decreased network's lifetime. Unlike BTA algorithm, the energy management of our proposal takes advantage from the ability of a sensor node to exchange fewer load information messages upon local load imbalance detection. This leads to lesser energy consumption and increased lifetime of the network.

To summarize, this study shows that, when dealing with large sensor networks, iterative local average decisions play an important role on the achieved global performances, namely the convergence time, the network's energy consumption, and the induced communication costs of the distributed consensus process.

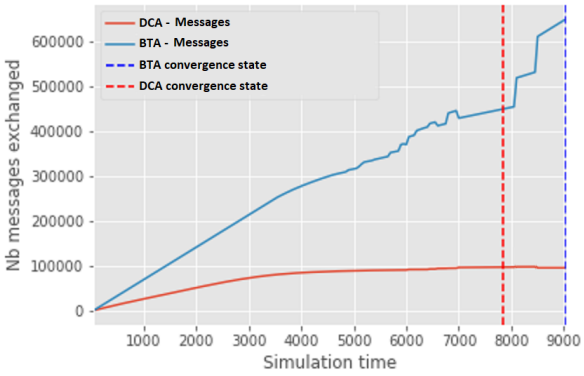
VII. Conclusion

In this paper, we have introduced a new fast asynchronous algorithm for distributed average consensus in WSN. Our proposal searches greedily for local imbalanced situations between neighbors and tries to ensure local equilibrium at each time-step of the average consensus process. For this end, instead of sending parsimonious load values from highly loaded nodes to less loaded ones, we move a large amount of load by performing parallel atomic transactions between pairs of neighbors. This improves the algorithm's convergence time speedup with low-cost communication and minimal energy consumption. Based on NS3 simulator, numerical results clearly show that our proposal outperforms its direct competitor due to *Bertsekas and Tsitsiklis* in all the tested cases. The obtained results reveal that local average decisions have a great impact on the global performances of the involved algorithms.

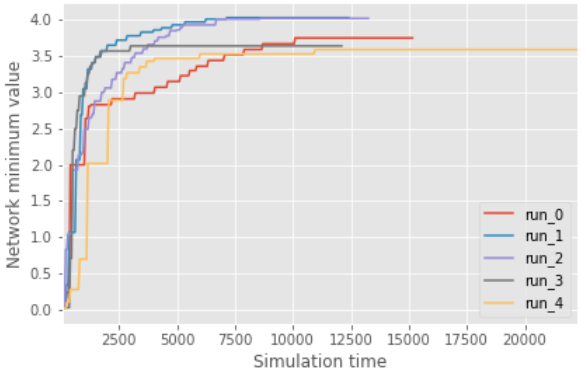
Extending this work for dependability issues in WSN will be an interesting research direction. Indeed, resource failures may occur and have a negative effect on the network's QoS, leading to inaccurate global average consensus values. We expect a difficult challenging trade-off between fault tolerance, convergence time, and energy consumption.



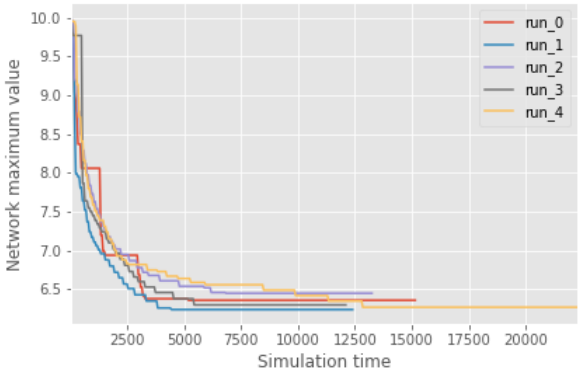
(a) Maximum delta convergence



(b) Number of exchanged messages

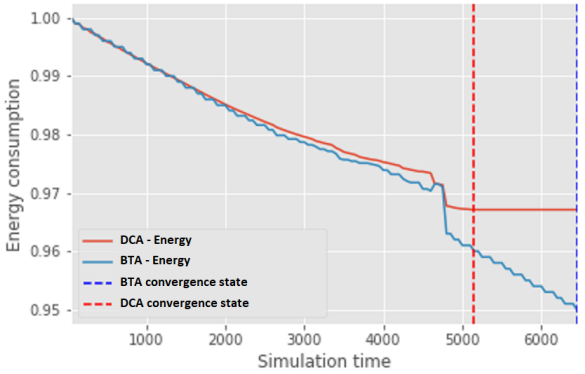


(c) Network minimum value

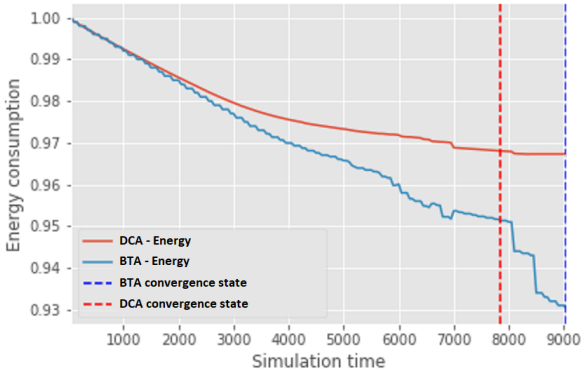


(d) Network maximum value

Fig. 3: Experimentation results with 600 nodes



(a) Experimental results with 100 nodes



(b) Experimental results with 600 nodes

Fig. 4: Energy consumption

References

[1] Jacques M. Bahi, Sylvain Contassot-Vivier, and Arnaud Giersch. Load balancing in dynamic networks by bounded delays asynchronous diffusion. In *High Performance Computing for Computational Science - VECPAR 2010 - 9th International conference, Berkeley, CA, USA, June 22-25, 2010, Revised Selected Papers*, pages 352–365, 2010.

[2] Jacques M. Bahi, Arnaud Giersch, and Abdallah Makhoul. A scalable fault tolerant diffusion scheme for data fusion in sensor networks. In *Infoscale 2008, The Third International ICST Conference on Scalable Information Systems*, page 10 (5 pages), Vico Equense, Italy, June 2008.

[3] Jacques M. Bahi, Christophe Guyeux, Mourad Hakem, and Abdallah Makhoul. Epidemiological approach for data survivability in unattended wireless sensor networks. *J. Network and Computer Applications*, 46:374–383, 2014.

[4] Jacques M. Bahi, Mohammed Haddad, Mourad Hakem, and Hamamache Kheddouci. Self-stabilizing consensus average algorithm in distributed sensor networks. *Trans. Large-Scale Data- and Knowledge-Centered Systems*, 9:28–41, 2013.

[5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[6] F. Bénézit. *Distributed Average Consensus for Wireless Sensor Networks*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2009.

[7] Ferran Cedó, Ana Cortés, Ana Ripoll, Miquel A. Senar, and Emilio Luque. The convergence of realistic distributed load-balancing algorithms. *Theory of Computing Systems*, 41(4):609–618, December 2007.

[8] Han-Lim Choi, Luc Brunet, and Jonathan P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. Robotics*, 25(4):912–926, 2009.

[9] Ana Cortés, Ana Ripoll, F. Cedo, Miquel A. Senar, and Emilio Luque. An asynchronous and iterative load balancing algorithm for discrete load model. *J. Parallel Distrib. Comput.*, 62(12):1729–1746, 2002.

[10] George Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.*, 7(2):279–301, 1989.

[11] Patrick Denantes, Florence Bénézit, Patrick Thiran, and Martin Vetterli. Which distributed averaging algorithm should I choose for my sensor network? In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, pages 986–994, 2008.

[12] M. Drozdowski. *Selected Problems of Scheduling Tasks in Multiprocessor Computer Systems*. PhD thesis, Poznan University of Technology, Poznan, Poland, 1998.

[13] Johannes Gehrke, C. Greg Plaxton, and Rajmohan Rajaraman. Rapid convergence of a local load balancing algorithm for asynchronous rings. *Theor. Comput. Sci.*, 220(1):247–265, 1999.

[14] Soummya Kar and José M. F. Moura. Distributed average consensus in sensor networks with random link failures. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, Honolulu, Hawaii, USA, April 15-20, 2007*, pages 1013–1016, 2007.

[15] Soummya Kar and José M. F. Moura. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. *IEEE Trans. Signal Processing*, 57(1):355–369, 2009.

[16] Soummya Kar and José M. F. Moura. Distributed consensus algorithms in sensor networks: quantized data and random link failures. *IEEE Trans. Signal Processing*, 58(3):1383–1400, 2010.

[17] J A. Legg. Tracking and sensor fusion issues in the tactical land environement. *Technical Report TN.0605*, 2005.

[18] Ciamac Cyrus Moallemi and Benjamin Van Roy. Consensus propagation. *IEEE Trans. Information Theory*, 52(11):4753–4766, 2006.

[19] R. Olfati-Saber. Distributed kalman filter with embeded consensus filters. *44th IEEE Conf. on Dec. and Cont.*, 2005.

[20] R. Olfati-Saber and J. S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. *44th IEEE Conf. on Dec. and Cont. CDC-ECC*, 2005.

[21] Reza Olfati-Saber. Distributed kalman filtering for sensor networks. In *46th IEEE Conference on Decision and Control, CDC 2007, New Orleans, LA, USA, December 12-14, 2007*, pages 5492–5498, 2007.

[22] Reza Olfati-Saber, J. Alexander Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[23] Stacy Patterson, Bassam Bamieh, and Amr El Abbadi. Convergence rates of distributed average consensus with stochastic link failures. *IEEE Trans. Automat. Contr.*, 55(4):880–892, 2010.

[24] Silvana Silva Pereira and Alba Pagès-Zamora. Mean square convergence of consensus algorithms in random wsns. *IEEE Trans. Signal Processing*, 58(5):2866–2874, 2010.

[25] Valentin Schwarz and Gerald Matz. Distributed reconstruction of time-varying spatial fields based on consensus propagation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2010, 14-19 March 2010, Sheraton Dallas Hotel, Dallas, Texas, USA*, pages 2926–2929, 2010.

[26] Mahendra Talasila, Shengli Fu, and Yan Wan. Energy conservative distributed average consensus through connected dominating set. In *2015 IEEE Wireless Communications and Networking Conference, WCNC 2015, New Orleans, LA, USA, March 9-12, 2015*, pages 843–848, 2015.

[27] Mohammad Sadegh Talebi, Mahdi Kefayati, Babak Hossein Khalaj, and Hamid R. Rabiee. Adaptive consensus averaging for information fusion over sensor networks. In *IEEE 3rd International Conference on Mobile Adhoc and Sensor Systems, MASS 2006, 9-12 October 2006, Vancouver, BC, Canada*, pages 562–565, 2006.

[28] Lin Xiao and Stephen P. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.

[29] Lin Xiao, Stephen P. Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *J. Parallel Distrib. Comput.*, 67(1):33–46, 2007.

[30] Lin Xiao, Stephen P. Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, April 25-27, 2005, UCLA, Los Angeles, California, USA*, pages 63–70, 2005.

[31] Lin Xiao, Stephen P. Boyd, and Sanjay Lall. A space-time diffusion scheme for peer-to-peer least-squares estimation. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks, IPSN 2006, Nashville, Tennessee, USA, April 19-21, 2006*, pages 168–176, 2006.

[32] Meng Zheng, Mario Goldenbaum, Slawomir Stanczak, and Haibin Yu. Fast average consensus in clustered wireless sensor networks by superposition gossiping. In *2012 IEEE Wireless Communications and Networking Conference, WCNC 2012, Paris, France, April 1-4, 2012*, pages 1982–1987, 2012.