Chapter 1
Distributed Average Consensus in Large Asynchronous Sensor Networks

Jacques M. Bahi, Arnaud Giersh, Abdallah Makhoul and Ahmed Mostefaoui

Abstract One important issue in sensor networks is parameters estimation based on nodes measurements. Several approaches have been proposed in the literature (centralized and distributed ones). Because of the particular noisy environment, usually observed in sensor networks, centralized approaches are not efficient and present several drawbacks (important energy consumption, routing information maintaining, etc.). In distributed approaches however, nodes exchange data with their neighbours and update their own data accordingly until reaching convergence to the right parameters estimate. These approaches, although provide some robustness against nodes failure, does not address important issues as communication delay tolerance and asynchronism (i.e., they require that nodes remain synchronous in communication and processing). In this chapter, we tackle these issues by proposing a totally asynchronous scheme that is communication delay tolerant. The extensive simulations series we conducted have showed the effectiveness of our approach.

1.1 Introduction

Recent years have witnessed significant advances in wireless sensor networks which emerge as one of the most promising technologies for the 21^{st} century [1]. In fact, they present huge potential in several domains ranging from health care applications to military applications. In general, the primary objective of a wireless sensor network is to collect data from the monitored area and to transmit it to a base station (sink) for processing. Many applications envisioned for sensor networks consist of lowpower and lowcost nodes. For instance, applications such as data fusion and distributed coordination require distributed function computation/parameter estima-

Computer Science Laboratory, University of Franche-Comté (LIFC)
Rue Engel-Gros, BP 527
90016 Belfort Cedex, France
e-mail: {firstname.lastname}@univ-fcomte.fr

tion under topology changes and power constraints. Distributed average consensus, in ad hoc networks, is an important issue in distributed agreement and synchronization problems [2] and is also a central topic for load balancing (with divisible tasks) in parallel computers [3]. More recently, it has also found applications in distributed coordination of mobile autonomous agents [4, 5] and distributed data fusion in sensor networks [6, 7]. In this chapter, we focus on a particular class of iterative algorithms based on information diffusion for average consensus, widely used in the applications cited above. Each node broadcasts its data to its neighbours and updates its estimation according to a weighted sum of the received data.

To illustrate the average consensus problem, let us consider the example of petrol tanks. We suppose that in a oil station we have large number of tanks related to each other in mechanical and sensor networks. The role of sensors is to dectect the level of each oil tank. The objective of this application is to keep the level of oil the same in all tanks. When a sensor node detects some changes in its level, it launchs an average consenus processus to calculate the average level of all tranks and then thanks to the mechanical network an oil transfer operation is done to regulate the level. The average consensus process is used to compute the average level, each sensor node exchanges its information with its neighbors by iterative manner until the convergence to the average consensus.

To calculate the average consensus, many distributed approaches have been proposed. On the other hand, these existing approaches present some insufficiencies (see next section). For instance, the flooding approach requires that each node holds a relatively important storage space. Other approaches make the unpractical assumption of communication synchronization between sensors [8, 5] and do not tolerate communication delays neither nodes failures. These weaknesses remain very restrictive in sensor network environment where on one hand nodes are prone to frequent failures as they are driven by batteries and on the other hand communications are almost unreliable and prone to delays. Moreover, these two limitative features lead, in addition to nodes mobility, to dynamically changing network topologies.

In order to overcome the above mentioned weaknesses, we propose and investigate in this chapter a novel approach for data fusion in sensor networks. The key idea behind is to develop a consensus algorithm that allows all nodes of the sensor network to track the average of their previous measurements [6, 8, 5, 9, 10, 11, 12, 13, 14]. More specifically, our proposition is based on an *in-network asynchronous iterative algorithm*, run by each node and in which nodes communicate with only their immediate neighbours.

In this context, let us discuss the primary contributions of this chapter:

- Our approach does not require any synchronization between nodes as it is basically asynchronous. In other words, each node communicates its data to its instantaneous neighbours at its own ”rhythm” i.e., no delays between nodes are observed in our approach. This is particularly important because in the synchronous schemes, as the one reported in [8], any delay between two nodes in the network will result in a global delay over the whole network since all the nodes are synchronous. This is particularly limitative in heterogeneous sensor networks where nodes have different processing speeds.

- As a consequence of its asynchronism, our proposed approach totally tolerates communication delays. This feature is of an important matter because sensor networks, as it is commonly known, are prone to environmental perturbations [15] when communication delays occur more frequently.
- The proposed distributed algorithm, as proven theoretical and validated experimentally, supports dynamic topologies and guarantees that each sensor node will converge to the average consensus.

However, as for any iterative approach, our approach could, under certain environmental conditions, consume more network resources, mainly communications, than other centralized approaches, specifically in ”perfect environment” where nodes and communications are totally reliable and the network topology is fixed. Nevertheless, we note here that our concern is more focused on *”noisy environment”* in which communication unreliability and nodes failures are usual.

1.2 Overview of Averaging Problem in Sensor Networks

The first and the simplest approach for distributed average estimation in sensor networks is called *flooding* approach [8]. In this approach, each sensor node broadcasts all its stored and received data to its neighbours. After a while, each node will hold all the data of the network and acts as a fusion center to compute the estimate of the unknown parameter. This technique has however several disadvantages [8]. First, it results in huge amount of exchanged duplicate messages, which represents a real limitation in environments like sensor networks. Second, flooding requires that each node stores at least one message per node (in order to compute the average). This could lead to an important storage memory requirement in case of a large sensor network with the associated operations (reads and writes). Finally, it is obvious that those requirements will consume much resources leading to an important decrease of the whole network lifetime.

Alternatively, in [16] the authors proposed a scalable sensor fusion scenario that performs fusion of sensor measurements combined with local Kalman filtering. They developed a distributed algorithm that allows the sensor nodes to compute the average of all of their measurements. It is worthy to note that many other sensor data fusion approaches are based on Kalman filters and mobile agents [17, 9, 18, 12, 13].

An iterative method for distributed data fusion in sensor networks based on the calculation of an average consensus [1] has been proposed in [8]. The authors consider that every node takes a noisy measurement of the unknown parameter. Each node broadcasts its data to its neighbours and updates its estimation according to a weighted sum of the received data. In this scheme all the communications are direct ones.

[1] In the rest of the paper, the terms ”average consensus” and ”parameter estimation” are used to denote the same mechanism of finding an estimate of the unknown parameter average.

Although the above mentioned works and other existing data fusion scenarios guarantee some level of robustness to nodes failures and dynamic topology changes [8, 16, 17, 11, 5], they either put some unpractical assumptions like nodes synchronization or do not support practical issues as the communication delays.

To the best of our knowledge, the above issues which are extremely important, especially in noisy environments, are not taken into account in previous data fusion approaches. In this chapter, we present an asynchronous data fusion scheme, particularly tailored to perturbed sensor networks. It focuses on a distributed iterative algorithm for calculating averages over asynchronous sensor networks. The sensor nodes exchange and update their data by the mean of a weighted sum in order to achieve the average consensus. The suggested algorithm does not rely on synchronization between the nodes nor does it require any knowledge of the global topology. To round up, the convergence of the proposed algorithm is proved in a general asynchronous environment.

1.3 Asynchronous Distributed Consensus with messages loss

1.3.1 Problem Formulation

A sensor network is modelled as a connected undirected graph $G = (V, E)$. The set of nodes is denoted by V (the set of vertices), and the links between nodes by E (the set of edges). The nodes are labelled $i = 1, 2, \dots, n$, and a link between two nodes i and j is denoted by (i, j). The dynamic topology changes are represented by the time varying graph $G(t) = (V, E(t))$, where $E(t)$ is the set of active edges at time t. The set of neighbours of node i at time t is denoted by $N_i(t) = \{j \in V \mid (i, j) \in E(t)\}$, and the degree (number of neighbours) of node i at time t by $\eta_i(t) = |N_i(t)|$.

Each node takes initial measurement z_i. For sake of simplicity let us suppose that $z_i \in \mathbb{R}$. Then, z will refer to the vector whose ith component is z_i in case we are concerned with several parameters. Each node on the network also maintains a dynamic state $x_i(t) \in \mathbb{R}$ which is initially set to $x_i(0) = z_i$.

Intuitively each node's state $x_i(t)$ is its current estimate of the average value $\sum_{i=1}^{n} z_i/n$. The goal of the averaging algorithm, is to let all the states $x_i(t)$ go to the average $\sum_{i=1}^{n} z_i/n$, as $t \to \infty$. This will be done through data exchange between neighbouring nodes where each node at every time iteration t performs weighted sum of the received data as follows [5, 8]:

$$x_i(t+1) = x_i(t) - \sum_{j \in N_i} \alpha_{ij}(t)(x_i(t) - x_j(t)), i = 1, \dots, n. \tag{1.1}$$

Where $\alpha_{ij}(t)$ is the weight on $x_j(t)$ at node i, and $\alpha_{ij}(t) = 0$ for $j \notin N_i(t)$.

In order to handle communication delays, we consider that at time t a node i gets the state of its neighbour j at time $d_j^i(t)$, where $0 \le d_j^i(t) \le t$

$d_j^i(t)$ represents the transmission delay between nodes i and j. Therefore, let us denote $x_j^i(t) = x_j(d_j^i(t)) \in \mathbb{R}$ the state of node j at time $d_j^i(t)$, received at time t by node i. Then, we defined the extended neighbourhood of node i at time t as the set:

$$\overline{N}_i(t) = \left\{ j \mid \exists\, d_j^i(t) \in \{t - B + 1, ..., t\} \text{, such that } j \in N_i(d_j^i(t)) \right\};$$

note that $N_i(t) \subset \overline{N}_i(t)$.

The problem, as for any distributed algorithmic approach, is how and under which conditions, will we ensure convergence of the proposed algorithm? In other terms, are we sure that all the node's x_i will converge to the right estimate of the unknown parameter average value? Also, how can we choose the parameters $\alpha_{ij}(t)$ so to improve the convergence speed and the quality of the derived estimate? Hereafter we present and analyse our proposal. We used the notations reported in Table 1.1

Notation	Description
$G(t)$	the time varying graph
$N_i(t)$	the set of neighbors of node i at time t
z_i	the initial measurement of node i
$x_i(t)$	the dynamic state of node i
$d_j^i(t)$	the transmission delay between nodes i and j
$x_j^i(t) = x_j(d_j^i(t))$	the state of node j at time $t - d_j^i(t)$
$\overline{N}_i(t)$	the extended neighborhood of i at time t
$s_{ij}(t)$	the data sent by i to j at time t
$r_{ji}(t)$	the data received by i from j at time t

Table 1.1 Notations

1.3.2 Asynchronous scheme

Our algorithm to compute the average consensus over the network is based on information diffusion i.e., each node takes a measurement and then cooperates with its neighbours in a diffusion manner to estimate the average of all the collected information. It is inspired from the work of Bertsekas and Tsitsiklis [19, section 7.4] on load balancing and extends it to cope with dynamic topologies and messages loss and delays. Algorithm 1 presents the main steps of our proposed algorithm.

Algorithm 1 The General Algorithm.

```
1: Each node maintains an instantaneous state $x_i(t) \in \mathbb{R}$, and at $t = 0$ (after all nodes have taken
   the measurement), each node initializes its state as $x_i(0) = z_i$.
2: At every step $t$ each node $i$:
   • compares its state to the states of its neighbours;
   • chooses and computes $s_{ij}(t)$. They have to be chosen carefully in order to ensure the
     convergence of the algorithm;
   • diffuses its information;
   • receives the information sent by its neighbours $r_{ji}(t)$;
   • updates its state with a combination of its own state and the states at its instantaneous and
     extended neighbours ($\overline{N}_i(t)$) as follows:
       $x_i(t+1) = x_i(t) - \sum_{j \in N_i(t)} s_{ij}(t) + \sum_{j \in \overline{N}_i(t)} r_{ji}(t).$   (1.2)
```

1.3.3 Theoretical Analysis (Convergence)

We now introduce three assumptions that ensure the convergence of our algorithm.

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

This assumption, known as jointly connected condition [8, 20], implies that each node i is connected to a node j within any time interval of length B and that the delay between two nodes cannot exceeds B. Recall that, a graph is connected if for any two vertices i and j there exists a sequence of edges $(i,\ k_1), (k_1, k_2), \ldots, (k_{l-1},\ k_l)$, $(k_l,\ j)$.

In Figure 1.1 we show an example of jointly connected graphs, we notice that at $t = 1$ the graph G_1 is not connected; the same case for G_2 at $t = 2$; while the union G of G_1 and G_2 is a connected graph.

Assumption 2 *There exists* $\alpha > 0, \forall t \geqslant 0$,
$\forall i \in N, \forall j \in N_i(t)$, *such that* $\alpha(x_i(t) - x_j^i(t)) \leq s_{ij}(t)$.
$\big(s_{ij}(t) = 0$ *if* $(x_i(t) \leq x_j^i(t))$ *for all* $j \in N_i(t)\big)$.

The second assumption postulates that when a node i detects a difference between its state and the states of its neighbours, it therefore computes non negligible s_{ij} to all nodes j where $(x_i(t) > x_j^i(t))$.

Assumption 3

$$x_i(t) - \sum_{k \in N_i(t)} s_{ik}(t) \geq x_j^i(t) + s_{ij}(t) \tag{1.3}$$

The third assumption prohibits node i to compute very large s_{ij} which creates a ping-pong state. Recall that, the ping-pong state is established when two nodes keep

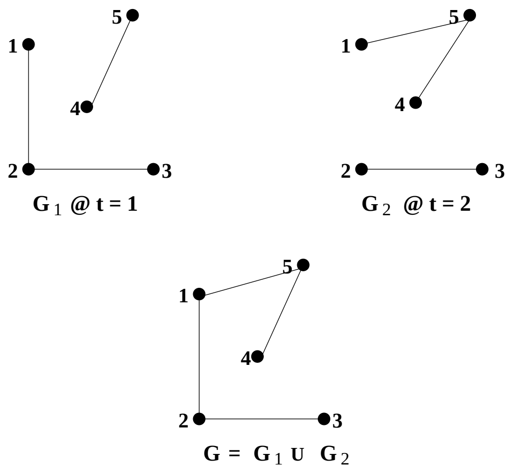

Fig. 1.1 Example of jointly connected graphs

sending data to each other back and forth, without ever reaching equilibrium. Note that these two assumptions are similar to assumption 4.2 introduced in [19, section 7.4].

Theorem 1. *if the assumptions 1, 2 and 3 are satisfied, Algorithm 1 guarantees that*

$$\lim_{t\to\infty} x_i(t) = \frac{1}{n}\sum_{i=1}^{n} x_i(0) \tag{1.4}$$

i.e., all node states converge to the average of the initial measurements of the network.

Proof

Let $m(t) = \min_i \min_{t-B<\tau\leq t} x_i(\tau)$. Note that $x^i_j(\tau) \geq m(t), \forall i, j, t.$
Lemma 1 and 2 below can be proven similarly to the lemma of pages 521 and 522 in [19].

Denote by $v_{ij}(t) = \sum\limits_{s=0}^{t-1} (s_{ij}(s) - r_{ij}(s))$, the data sent by i and not yet received by j at time t. We suppose that $v_{ij}(0) = 0$. Then by data conservation, we obtain

$$\sum_{i=1}^{n}\left(x_i(t) + \sum_{j\in N_i(t)} v_{ij}(t)\right) = \sum_{i=1}^{n} x_i(0), \qquad \forall t \geqslant 0 \tag{1.5}$$

¿From assumption 1 we can conclude that the data $v_{ij}(t)$ in the network before time t consists in data sent in the interval time $\{t-B+1, ..., t-1\}$, so $v_{ij}(t) \leq \sum_{\tau=t-B+1}^{t-1} s_{ij}(t), \forall nodei, \forall j \in N_i(t).$

Lemma 1. *The sequence $m(t)$ is monotone, nondecreasing and converges and* $\forall i, \forall s \geq 0,$

$$x_i(t+s) \geq m(t) + \left(\frac{1}{n}\right)^{t_1 - t_0} (x_i(t) - m(t))$$

Let $i \in V, t_0 \in \mathbb{N}$, and $t \geq t_0$, $j \in V$, we say that the event $E_j(t)$ occurs if there exists $j \in \overline{N}_i(t)$ such that

$$x_j^i(t) < m(t_0) + \frac{\alpha}{2n^{t-t_0}} (x_i(t_0) - m(t_0)) \tag{1.6}$$

and

$$s_{ij}(t) \geq \alpha \left(x_i(t) - x_j^i(t)\right), \tag{1.7}$$

where α is defined in assumption 2, and V is the set of all nodes.

Lemma 2. *Let $t_1 \geq t_0$, if $E_j(t_1)$ occurs, then $E_j(\tau)$ doesn't occur for any $\tau \geq t_1 + 2B$.*

Lemma 3. $\forall i \in V, \forall t_0 \in \mathbb{N}, \forall j \in \overline{N}_i(t)$,

$$t \geq t_0 + 3nB \Rightarrow x_j(t) \geq m(t_0) + \eta \left(\frac{1}{n}\right)^{t-t_0} (x_i(t_0) - m(t_0)).$$

where $\eta = \frac{\alpha}{2}\left(\frac{1}{n}\right)^B$.

Proof. Let us fix i and t_0. Let us consider $t_1, ..., t_n$ such that $t_{k-1} + 2B \leq t_k \leq t_{k-1} + 3B$. Lemma 2 implies that if $k \neq l$, then $E_j(t_k)$ and $E_j(t_l)$ doesn't occur together. Hence, there exists t_k for which (1.6) is not satisfied for all $d_j^i(t_k) \in \{t_k - B + 1, ..., t_k\}$, and $j \in N_i(d_j^i(t_k))$.

Let $j^* \in N_i(d_j^i(t_k))$ such that $x_{j^*}^i(t_k) \leq x_j^i(t_k), \forall j \in N_i(d_j^i(t_k))$. Since (1.6) is not satisfied for $j = j^*$, we have

$$\begin{aligned} &x_j^i(t_k) \geq x_{j^*}^i(t_k) \\ &x_{j^*}^i(t_k) \geq m(t_0) + \tfrac{\alpha}{2}\left(\tfrac{1}{n}\right)^{t_k - t_0}(x_i(t_0) - m(t_0)),\ \forall j \in N_i(d_j^i(t_k)). \end{aligned}$$

For $t \geq t_0 + 3nB$, we have $t \geq t_k \geq d_j^i(t_k)$. Lemma 1 gives, $\forall j \in N_i(d_j^i(t_k))$

$$\begin{aligned} x_j(t) &\geq m(d_j^i(t_k)) + \left(\tfrac{1}{n}\right)^{t - d_j^i(t_k)} (x_j(d_j^i(t_k)) - m(d_j^i(t_k))) \\ &\geq m(t_0) + \tfrac{\alpha}{2}\left(\tfrac{1}{n}\right)^B \left(\tfrac{1}{n}\right)^{t-t_0} (x_i(t_0) - m(t_0)). \end{aligned}$$

Definition 1. We say that a sensor j is l-connected to a sensor i if it is logically connected to i by l communication graphs, i.e. if there exists $r_k^i(t_k) \in \{t_k - B + 1, ..., t_k\}$, where $k \in \{i_1, ..., i_l\}$, such that $i = i_1 \in N_{i_2}(r_{i_1}^{i_2}(t_1)), i_2 \in N_{i_3}(r_{i_2}^{i_3}(t_2)), ...,$
$i_l \in N_j(r_l^j(t_l))$.

Lemma 4. *If sensor j is l-connected to sensor i then*

$$\forall t \geq t_0 + 3nlB,\ x_j(t) \geq m(t_0) + (\eta)^l \, (\frac{1}{n})^{(t-t_0)^l} \left(x_i(t_0) - m(t_0)\right).$$

Proof. By induction. Suppose that the lemma is true for $t_0 + 3nlB$ then if j is l-connected to j, we have

$$x_l(t_0 + 3nlB) \geq m(t_0) + (\eta)^l \left((\frac{1}{n})^{(3nlB)} \right)^l \left(x_i(t_0) - m(t_0)\right).$$

Consider a sensor k connected to j (k is $(l+1)$-connected to i), Lemma 3 and the above inequality give (replacing t_0 by $t_0 + 3nlB$),

$$\begin{aligned} &x_k(t) \geq \\ m(t_0 + 3nlB) + \eta(\tfrac{1}{n})^{t-t_0-3nlB} &\left((\eta)^l \left((\tfrac{1}{n})^{(3nlB)} \right)^l \left(x_i(t_0) - m(t_0)\right) \right) \\ &\geq \\ m(t_0) + (\eta)^{l+1} &\left((\tfrac{1}{n})^{(t-t_0)} \right)^{l+1} \left(x_i(t_0) - m(t_0)\right). \end{aligned}$$

Proof (Proof of Theorem 1). Consider a sensor i and a time t_0. Assumption 1 implies that sensor i is B-connected to any sensor j. Lemma 4 gives: $\forall t \in [t_0 + 3nMB, t_0 + 3nMB + B]\,, \forall j \in V$,

$$x_j(t_0 + 3nMB + B) \geq m(t_0) + \delta \left(x_i(t_0) - m(t_0)\right),$$

where $\delta > 0$. Thus,

$$m(t_0 + 3nMB + B) \geq m(t_0) + \delta \left(\max_i x_i(t_0) - m(t_0) \right).$$

Note that $\lim_{t_0 \to \infty} \max_i x_i(t_0) - m(t_0) = 0$ (otherwise $\lim_{t_0 \to \infty} m(t_0) = +\infty$). On the other hand, as $\lim_{t \to \infty} m(t) = c$ and as $m(t) \leq x_j(t) \leq \max_i x_i(t)$, we deduce that $\forall j \in V$, $\lim_{t \to \infty} x_j(t) = c$, which implies that $\lim_{t \to \infty} s_{ij}(t) = 0$. Thanks to assumption 1, we deduce that $\lim_{t \to \infty} v_{ij}(t) = 0$, and thanks to (1.5), we deduce that $nc = \lim_{t \to \infty} x_i(t) = \sum_{i=1}^n x_i(0)$,i.e. $c = \sum_{i=1}^n x_i(0)/n$, which yields to $\lim_{t \to \infty} x_i(t) = \frac{1}{n} \sum\limits_{i=1}^{n} x_i(0)$ proving Theorem 1.

1.3.4 Practical Issues

We now discuss some practical aspects related to the implementation of Algorithm 1. The main two points are how to choose $s_{ij}(t)$ and how to overcome the loss of messages?

Each node updates its state following equation (1.2). This is achieved, by updating each sensors $s_{ij}(t)$ through time. For sake of simplicity, the value of $s_{ij}(t)$ is chosen to be computed by the weighted difference between the states of nodes i and j as follows:

$$s_{ij}(t) = \begin{cases} \alpha_{ij}(t)(x_i(t) - x_j^i(t)) & \text{if} \quad x_i(t) > x_j^i(t) \ , \\ 0 & \text{otherwise.} \end{cases}$$

The choice of $s_{ij}(t)$ is then deduced from the proper choice of the weights $\alpha_{ij}(t)$. Hence, $\alpha_{ij}(t)$ must be chosen such that the states of all the nodes converge to the average $\sum_{i=1}^{n} z_i/n$, i.e., assumptions 2 and 3 must be satisfied.

Denote by j^* the sensor node satisfying $x_{j^*}^i = \min_{k \in N_i(t)} x_k^i(t)$ (note that j^* depends on i and time t). The values of $\alpha_{ij}(t)$ must be selected so that to avoid the ping pong condition presented in assumption 3.

This is equivalent to choose $\alpha_{ij}(t)$ so that $\forall t \geqslant 0, \forall i \in N$, and $j \neq j^* \in \overline{N}_i(t)$ satisfying $x_i(t) > x_j^i(t)$,

$$0 \leq \alpha_{ij}(t) \leq \frac{1}{2}\left(1 - \frac{\sum_{j \neq i} \alpha_{ik}(t)(x_i(t) - x_i^k(t))}{(x_i(t) - x_i^j(t))}\right) \tag{1.8}$$

The weights $\alpha_{ij}(t)$ must also be chosen in order to respect assumption 2. This assumption can be carried out by fixing a constant $\beta \in [0,1]$ and choosing

$$\begin{cases} \sum_{k \neq j^* \in N_i(t)} \alpha_{ik}(t)(x_i(t) - x_k^i(t)) \leq \beta(x_i(t) - x_{j^*}^i(t)), \\ \alpha_{ij^*}(t) = \frac{1}{2}\left(1 - \frac{\sum_{k \neq j^*} \alpha_{ik}(t)(x_i(t) - x_k^i(t))}{(x_i(t) - x_{j^*}^i(t))}\right) \end{cases} \tag{1.9}$$

Indeed, from (1.9) we deduce

$$\alpha_{ij^*}(t) \geq \frac{(x_i(t) - x_{j^*}^i(t)) - \beta(x_i(t) - x_{j^*}^i(t))}{2(x_i(t) - x_{j^*}^i(t))} = \frac{1-\beta}{2} = \alpha.$$

Hence, $\forall i, j^*, t$ such that $j^* \in \overline{N}_i(t)$ and $x_{j^*}^i(t) = \min_{k \in N_i(t)} x_k^i(t)$,

$$s_{ij^*}(t) = \alpha_{ij^*}(t)\left(x_i(t) - x_{j^*}^i(t)\right) \geq \alpha\left(x_i(t) - x_{j^*}^i(t)\right).$$

The first inequation of (1.9) can be written as $\sum_{k \neq j^* \in V_i(t)} s_{ik}(t) \leq \beta(x_i(t) - x_{j^*}^i(t))$, this means that the totality of data sent to the neighbours of i (except j^*) doesn't exceed a portion β of $(x_i(t) - x_{j^*}^i(t))$.

Equations (1.8) and (1.9) are derived from the assumptions 2 and 3. Therefore the choice of the weights α_{ij} must take into consideration these two equations.

First let define the deviation $\Delta_i^j(t)$ of node i as:

$$\Delta_i^j(t) = \begin{cases} x_i(t) - x_j^i(t) & \text{if} j \in N_i(t) \text{ and } x_i(t) > x_j^i(t) \ , \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 2 Temporally updating weights of node i.

```
1: for $j \leftarrow 1$ to $n$ do
2:     if $j \neq i$ then
3:         $s_{ij} \leftarrow 0$
4:         $\alpha_{ij} \leftarrow 0$
5:     end if
6: end for
7: $k \leftarrow 0$
8: $Sum \leftarrow 0$
9: find $\ell$ such that $\Delta_i^\ell = Delta_i[k]$
10: $\alpha_{i\ell} = 1/(\eta_i + 1)$
11: $s_{i\ell} = \alpha_{i\ell} \times \Delta_i^\ell$
12: repeat
13:     $Sum \leftarrow Sum + s_{il}$
14:     $k \leftarrow k + 1$
15:     find $\ell$ such that $\Delta_i^\ell = Delta_i[k]$
16:     $\alpha_{i\ell} \leftarrow 1/(\eta_i + 1)$
17:     $s_{i\ell} \leftarrow \alpha_{i\ell} \times \Delta_i^\ell$
18: until $NOT\ ((x_i - Sum \geq x_\ell^i + s_{i\ell})\ AND\ (k < n))$
```

Algorithm 2 presents our method for temporally updating the averaging weights. Node i computes the difference between its current state and current states of its neighbours. The positive deviations ($\Delta_i^j > 0$) are then stored in the array $Delta_i$, in a decreasing order. Then, it sets the weight α_{ij} to $1/(\eta_i(t) + 1)$, where $\eta_i(t)$ is the current number of its neighbours, starting by its neighbours nodes j whose have the larger deviations while respecting assumption 3.

In order to cope with the problem of message loss, we adopted the following strategy: instead of sending $s_{ij}(t)$ from node i to node j, it is the sum $\Sigma_{s_{ij}}(t) = \sum_{0 \leq \tau \leq t} s_{ij}(\tau)$ that is sent. Symmetrically the receivers maintains the sum of the received data $\Sigma_{r_{ji}}(t) = \sum_{0 \leq \tau \leq t} r_{ji}(\tau)$. Upon receiving, at a time t, a message from node i, a node j can now recover all the data that was sent before time $d_i^j(t)$. It has only to calculate the difference between the received $\Sigma_{s_{ij}}(d_i^j(t))$ and the locally stored $\Sigma_{r_{ji}}(t)$.

To conclude, the state messages exchanged during the execution of the algorithm are composed of two scalar values : the current state of the node, $x_i(t)$, and the sum of the sent data $\Sigma_{s_{ij}}(t)$.

1.3.5 Illustrative Example

To illustrate the behaviour of our proposed approach, les us consider the example presented in Figure 1.2. It consists in a network of four nodes. The initial measurement of each node i is known by z_i and the initial state $x_i(0) = z_i$.

Following the second step of Algorithm 1, each node computes the weights α_{ij} for its neighbours. This is done by using Algorithm 2.

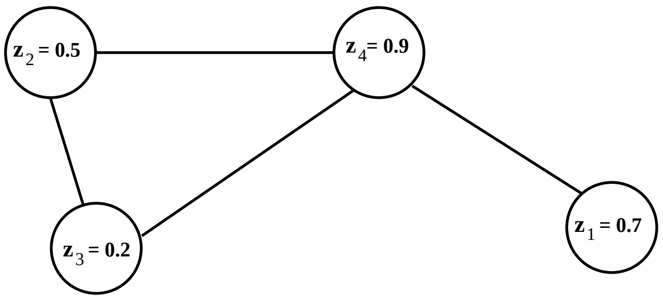

Fig. 1.2 An example of a sensor network composed of four nodes with their initial measurements.

Let us focus on the case of $node_4$ for instance. We notice that it has three neighbours and the high deviation Δ corresponds to $node_3$. Therefore, it computes $\alpha_{43}(0) = \frac{1}{\eta_4+1} = \frac{1}{4}$ first, such that η_4 is the number of its neighbours. Then, $s_{43}(0) = \frac{1}{4}(x_4(0) - x_3(0)) = 0.175$. For the two reminder neighbours $node_1$ and $node_2$, $node_4$ computes $\alpha_{42}(0)$ first for the reason that Δ_4^2 is higher than Δ_4^1. We note that for $node_2$ the Assumption 3 (ping pong condition) is satisfied while it is not the case for $node_1$ which leads to $\alpha_{41}(0) = 0$.

All the nodes compute their weights and then diffuse their information to their neighbours to update their states following Equation (1.2). For the above example after the first step we obtain:

$$x_1(1) = 0.7$$
$$x_2(1) = 0.5 + 0.1 - 0.1 = 0.5$$
$$x_3(1) = 0.2 + 0.1 + 0.175 = 0.475$$
$$x_4(1) = 0.9 - 0.1 - 0.175 = 0.625$$

This process is repeated for several iterations until all the states of the nodes converge to the average of the initial measurements. We note that our scheme is robust to the topology changes and the loss of messages as discussed in details in the next section.

1.4 Experimental Results

In order to evaluate the performance of our approach, we have implemented a simulation package using the discrete event simulator OMNET++ [21]. This package includes our asynchronous algorithm as well as a synchronous one. As confirmed in previous related works [8, 5], distributed approaches out perform centralised approaches, in particular in noisy networks. For this reason, we have not included in our comparison centralised approaches, and focuses instead on synchronous distributed approaches that are more closed to our work.

We performed several runs of the algorithms (an average of 100 runs). In each experimental run, the network graph is randomly generated, where the nodes are distributed over a $[0, 100] \times [0, 100]$ field. The node communication range was set to 30. The initial node measurements z_i were also randomly generated. Each node is aware of its immediate neighbours through a ”hello” message. Once the neighbourhood is identified, each node run the algorithm i.e., begins exchanging data until convergence.

We studied the performance of our algorithm with regard to the following parameters:

- Robustness in front of communication failures: we mainly varied the probability of communication failure, noted p. This parameter allows us to highlight the behaviour of our scheme in noisy environment and in dynamic topologies.
- Scalability: we varied the number of sensor nodes deployed in the same area to see how our proposed approach scales?

The main metrics we measured in this paper are: (a) the mean error between the current estimate x_i and the average of the initial data, (b) the mean number of iterations necessary to reach convergence and (c) the overall time before reaching the global convergence. We note here that in asynchronous algorithms, there is no direct correlation between the number of iterations and the total time to convergence, contrary to synchronous approaches. In fact, as there are no delays between nodes, the number of iterations could be relatively high. This does not mean that the total time to convergence could be long too. For this reason, we have made the distinction between the number of iterations and the time taken to reach convergence. As we run a discrete event simulation package, this time is the one given by the discrete simulator OMNET++ [21]; we named it *simulated time*. For all the experiments, the global convergence state is said to be reached when $\varepsilon_i = |x_i - \sum_{i=1}^{n} y_i/n|$ becomes less than some fixed constant ε.

Note that, in the figures next sections, the points represent the obtained results and the curves are an extrapolation of these points.

1.4.1 Basic Behaviour

First, we show simulation results for the case where we have a fixed topology with a fixed number of nodes (50 nodes) and $\varepsilon = 10^{-4}$. The mean error of the nodes $\varepsilon' = \sum_{i=1}^{n} \varepsilon_i/n$ was plotted in Figure 1.3. As expected, it can be seen that the convergence in the synchronous mode is faster than the convergence in the asynchronous one. It is also noticed that the two graphs have the same pace.

However, in many scenarios an exact average is not required, and one may be willing to trade precision for simplicity. For instance, minimizing the number of iterations to reduce the energy consumption can be privileged in sensor networks applications where exact averaging is not essential.

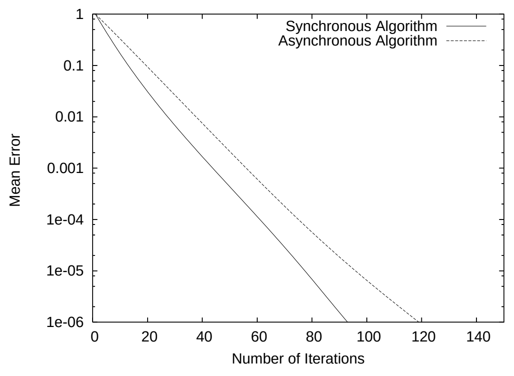

Fig. 1.3 The Mean Error ε

1.4.2 Dynamic topology

In a next step, we simulated the proposed sensor fusion scheme with dynamically changing communication graphs. We generated the sequence of communication graphs as follows: at each time step, each edge in the graph is only available with a selected probability p, independent of the other edges and all previous steps. To ensure the jointly connected condition of the generated graphs, we selected a period of time τ in which an edge cannot stay disconnected more than τ time.

We fixed the number of sensor nodes to 50 and $\varepsilon = 10^{-4}$. In preliminary results, the period τ was chosen in a way that is equal to three times the time of a communication. We show in figure 1.4 and figure 1.5 the variation of the number of iterations and the time simulation with the probability of link failure p. We notice that the number of iterations and the overall time increase with the increase of the probability, but not in an exponential way.

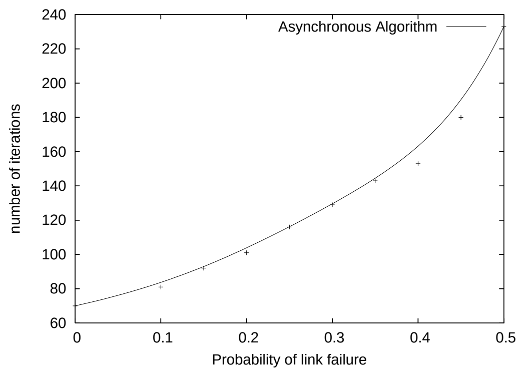

Fig. 1.4 Number of Iterations

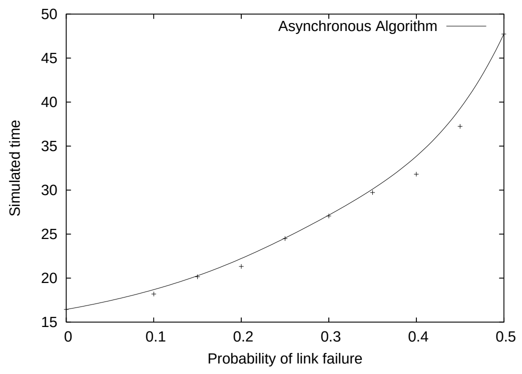

Fig. 1.5 Simulated Time

Note that we also tried to run the synchronous algorithm with dynamic topology changes, but the execution times were so prohibitive, that we abandoned those experiments. These results confirm that synchronous algorithms are infeasible for real sensor networks.

1.4.3 Larger Sensor Network

Our scheme can be applied to sensor networks where a large number of sensor nodes are deployed, since it is fully distributed and there is no centralized control. In our simulations we varied the number of sensor nodes from 20 to 200 nodes, deployed in the region $[0,100] \times [0,100]$, we selected for all nodes i, $\varepsilon = 10^{-4}$.

However, as shown in the two Figures (Figure 1.6 and Figure 1.7), as the number of sensor nodes increases, the average of the iterations number as well as the time needed to reach global convergence decreases in the two cases synchronous and asynchronous. We notice that in the synchronous mode we obtained less number of iterations, on the other hand it takes more time to reach the global convergence than the asynchronous one.

1.5 Further Discussions

In this section, we give further consideration to our data fusion scheme from the viewpoints of robustness to the delays and loss of messages and energy efficiency in comparison to other existing works.

Sensor nodes are small-scale devices. Such small devices are very limited in the amount of energy they can store or harvest from the environment. Thus, energy efficiency is a major concern in a sensor network. In addition, many thousands of

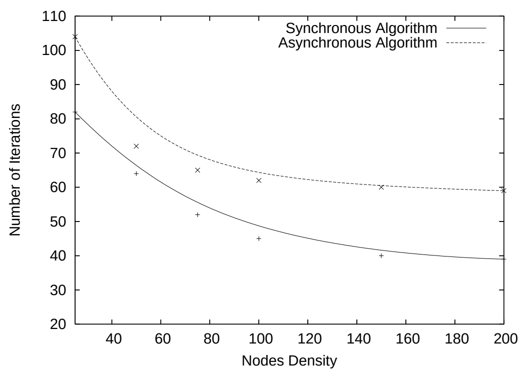

Fig. 1.6 Number of iterations

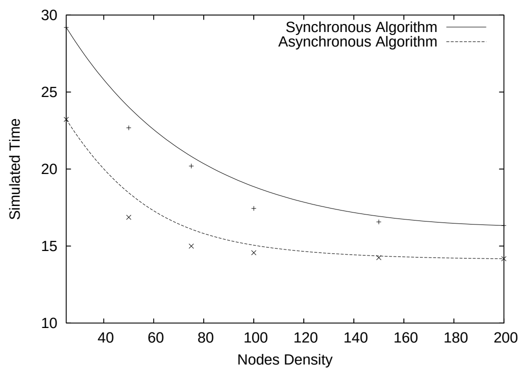

Fig. 1.7 Simulated Time

sensors may have to be deployed for a given task. An individual sensor's small effective range relative to a large area of interest makes this a requirement.

Therefore, scalability is another critical factor in the network design. Sensor networks are subject to frequent partial failures such as exhausted batteries, nodes destroyed due to environmental factors, or communication failures due to obstacles in the environment. Message delays can be rather high in sensor networks due to their typically limited communication capacity which is shared by nodes within communication range of each other. The overall operation of the sensor network should be robust despite such partial failures.

In our scheme, we presented a scalable asynchronous method for averaging data fusion in sensor networks. The simulations we conducted show that, the higher the density of the deployed nodes, the more the precise of the estimation would be. On the other hand, our algorithm is totally asynchronous, where we consider delay transmission and loss of messages in the proposed model. These aspects which are highly important are not taken into account in previous sensor fusion works [8, 14].

Another important practical issue in sensor network is the power efficiency. Optimizing the energy consumption in sensor networks is related to minimize the number of the network communications as the radio is the main energy consumer in a sensor node [1]. Considering the distributed iterative procedure for calculating averages, the only way to minimize the energy consumption is to reduce the number of iterations before attending the convergence. To show how well our algorithm saves energy, we compared our obtained results to those reported by another diffusive scheme for average computation in sensor networks [8]. For instance, in a static topology our algorithm converges after 69 iterations with a mean error of 10^{-4} while the best results in the second approach reached 85 iterations for the same mean error. For the dynamic topology mode, we obtained 105 iterations, mean error 10^{-4} and probability of link failure 0.25, while the number of iterations is very high (≈ 300 iterations) in [8].

1.6 Conclusion and Future Work

In this paper, we introduced a fault tolerant diffusion scheme for data fusion in sensor networks. This algorithm is based on data diffusion; the nodes cooperate and exchange their information only with their direct instantaneous neighbours. In contrast to existing works, our algorithm does not rely on synchronization nor on the knowledge of the global topology. We prove that under suitable assumptions, our algorithm achieves the global convergence in the sense that, after some iterations, each node has an estimation of the average consensus overall the whole network. To show the effectiveness of our algorithm, we conducted series of simulations and studied our algorithm under various metrics.

In our scenario, we have focused on developing a reliable and robust algorithm from the view points of asynchronism and fault tolerance in a dynamically changing topology. We have taken into account two points which don't have been previously addressed by other authors, namely the delays between nodes and the loss of messages. Knowing that in real sensor networks the nodes are prone to failures. One of the near future goals is to allow nodes to be dynamically added and removed during the execution of the data fusion algorithm. We also plan to test our algorithm in a real-world sensor network.

References

1. I. Akyildiz, W. Su, Y. Sankarasubramniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, pages 102–114, 2002.
2. N.A. Lynch. Distributed algorithms. *Morgan Kaufmann Publishers, Inc.*, 1996.
3. Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load-balancing schemes. *in Proceedings of the IEEE Symp. on Found. of Comp. Sci., Palo Alto*, 1998.

4. P.A. Bliman and G. Ferrari-Trecate. Average consensus problems in networks of agents with delayed communications. *Journal of IFAC*, 44(8):1985–1995, 2008.
5. R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transaction on Automatic Control*, 49(9):1520–1533.
6. Jacques Bahi, Arnaud Giersch, and Abdallah Makhoul. A scalable fault tolerant diffusion scheme for data fusion in sensor networks. *The Third International ICST Conference on Scalable Information Systems, Infoscale 2008, ACM*, june 2008.
7. C. C. Moallemi and B. V. Roy. Consensus propagation. *IEEE Trans. Inf. Theory*, 52(11):4753–4766, Nov 2006.
8. L. Xiao, S. Boyd, and S. lall. A scheme for robust distributed sensor fusion based on average consensus. *Proc. of the International Conference on Information processing in Sensor Networks (IPSN)*, pages 63–70, 2005.
9. A. Speranzon, C. Fischione, and K.H. Johansson. Distributed and collaborative estimation over wireless sensor networks. *Proceedings of 45th IEEE Conference on Decision and Control*, 2006.
10. L. Xiao, S. Boyd, and S. Lall. A space-time diffusion scheme for peer-to-peer least-squares estimation. *Proc. of Fifth International Conf. on Information Processing in Sensor Networks (IPSN 2006)*, pages 168–176, 2006.
11. Mohammad S. Talebi, Mahdi Kefayati, Babak H. Khalaj, and Hamid R. Rabiee. Adaptive consensus averaging for information fusion over sensor networks. *In the proceedings of The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'06)*, 2006.
12. D. Spanos, R. Olfati-Saber, and R.M. Murray. Distributed sensor fusion using dynamic consensus. *proceedings of IFAC*, 2005.
13. D.S. Scherber and H.C. Papadopoulos. Distributed computation of averages over ad hoc networks. *IEEE journal on Selected Areas in Communications*, 23(4):776–787, April 2005.
14. Mohammad S. Talebi, Mahdi Kefayati, Babak H. Khalaj, and Hamid R. Rabiee. Adaptive consensus averaging for information fusion over sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 562–565, 2006.
15. J A. Legg. Tracking and sensor fusion issues in the tactical land environement. *Technical Report TN.0605*, 2005.
16. R. Olfati-Saber and J. S. Shamma. Consensus filters for sensor networks and distributed sensor fusion. *Proceedings of 44th IEEE Conference on Decision and Control CDC-ECC*, 2005.
17. R. Olfati-Saber. Distributed kalman filter with embeded consensus filters. *Proceedings of 44th IEEE Conference on Decision and Control*, 2005.
18. R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proc. of IEEE*, pages 215–233, 2007.
19. Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.
20. Jacques Bahi, Raphael Couturier, and Flavien Vernier. Synchronous distributed load balancing on dynamic networks. *Journal of Parallel and Distributed Computing*, 65(11):1397–1405, 2005.
21. OMNeT++. http://www.omnetpp.org/.