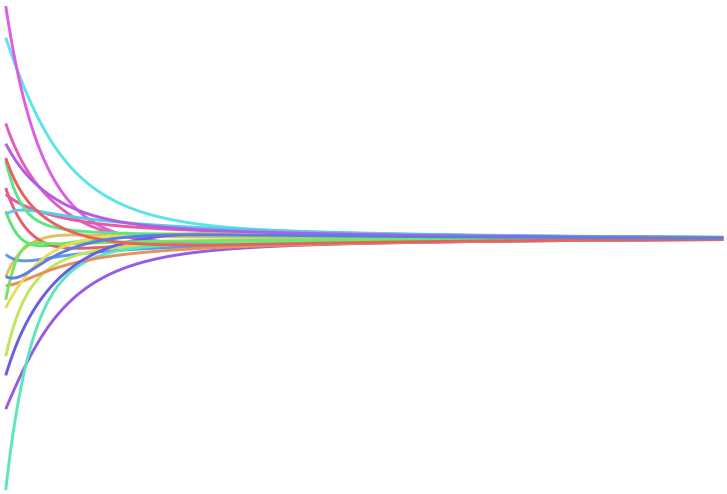
CHALMERS



Consensus Trade-offs in Wireless Sensor Networks

CHRISTOPHER LINDBERG

Communication Systems Group
Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

Thesis for the degree of Licentiate of Engineering

Consensus Trade-offs in Wireless Sensor Networks

Christopher Lindberg

CHALMERS

Communication Systems Group
Department of Signals and Systems
Chalmers University of Technology

Gothenburg, Sweden 2015

Lindberg, Christopher
Consensus Trade-offs in Wireless Sensor Networks

Department of Signals and Systems
Technical Report No. R011/2015
ISSN 1403-266X

Communication Systems Group
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Telephone: + 46 (0)31-772 1597
Email: chrlin@chalmers.se

This thesis has been prepared using LaTeX.

Printed by Chalmers Reproservice,
Göteborg, Sweden, November 2015.

To whom it may concern.

"A room without books is like a body without soul."

– Marcus Tullius Cicero

Abstract

As more and more everyday electronic devices become equipped with the combined resources of computation, sensing, and wireless communications, possible platforms for implementation of wireless sensor networks have become ubiquitous. The combination of these three main capabilities of such a network present the opportunity to for example gather high resolution measurement data, or cooperatively perform advanced computational tasks. The size of these wireless sensor networks are often said to be on the order of hundreds to several thousands nodes. To harness the capabilities of these networks, there is a need to design and implement efficient distributed algorithms.

In this thesis we study a family of distributed algorithms referred to as *consensus algorithms*. These algorithms work by nodes in the network locally exchanging information with the goal of reaching an agreement (consensus) on something. They then use the information received from their neighbors to update their information according to an update rule determined by the algorithm. This is done repeatedly until consensus on the information is reached, or we decide to stop the algorithm. Specifically, we study design choices, or trade-offs, that should be considered when implementing consensus algorithms to solve certain distributed problems in wireless sensor networks. The trade-offs investigated in this thesis and the appended papers deal with the metrics of time, communication resource usage, and information disagreement.

In the overview part of the thesis we give an introduction to some consensus algorithms, and show how the convergence properties can be analyzed using spectral analysis of matrices. We also briefly introduce two trade-offs, and give some intuition of the mechanics behind their behaviors. In the appended papers, we present an application of consensus algorithms to in-network compression using the theory of compressed sensing. We derive upper bounds for the reconstruction performance. These bounds are then used to formulate a trade-off between two communication costs, one cost for sensor-to-sensor communication and one for sensor-to-sink communication, given a reconstruction error threshold. Furthermore, we evaluate the optimal system design given the cost ratio of the two communication costs. We also use a consensus algorithm in combination with a distributed particle filter to perform distributed tracking. The trade-off considered for this scenario is one weighing tracking performance against time delay, with the design parameter being the communication range affecting both of these metrics.

Keywords: Consensus Algorithms, Wireless Sensor Networks, Trade-off analysis, Distributed algorithms, Compressed Sensing

List of Included Publications

The thesis is based on the following appended papers:

[A] C. Lindberg, L. S. Muppirisetty, K-M. Dahlén,V. Savic, and H. Wymeersch, "MAC Delay in Belief Consensus for Distributed Tracking," in *Proceedings of the 10th Workshop on Positioning, Navigation and Communication (WPNC)*, Dresden, Germany, Mar. 2013.

[B] C. Lindberg, A. Graell i Amat, and H. Wymeersch, "Distributed Compressed Sensing for Sensor Networks with Packet Erasures," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, pp. 13–19, Austin, TX, USA, Dec. 2014.

[C] C. Lindberg, A. Graell i Amat, and H. Wymeersch, "Compressed Sensing for Wireless Sensor Networks Without Explicit Position Information," in preparation.

Acknowledgements

My friend and colleague Erik Steinmetz told me as he was writing his thesis, that the acknowledgements is the hardest part to write, and I think he is right. There are so many people to thank, and so many ways to say it, but it is hard to find those perfect lines that capture everything. So, you'll have to settle for this:

First of all, I would like to thank my main supervisor Assoc. Prof. Henk Wymeersch and co-supervisor Assoc. Prof. Alexandre Graell i Amat for the motivation and guidance you have given me. It has not been an easy journey, and you have shown me so much patience, but it has been a lot of fun too. I don't think I really understood or believed in the concept of mentor before I started my PhD but now I can say that I do. Again, thank you for everything you have done for me.

I would also like to thank the head of the ComSys group at Chalmers, Prof. Erik Ström, for creating the most awesome workplace. I could not imagine a better place to work in! A special thanks to my buddy and office mate Srikar, thanks for always bringing some laughter with you to lighten up the day. Thanks to our own group within the group, the COOPNET: Rocco, Gabo, Erik, Markus, Robert, Jie, Themis, Hamed, Zhenhua, and Ali. I had many interesting discussions and seminars with all of you. Of course, special thanks to Themis for the amazing pastitio and other cypriot specialities. I'm counting on you for our next meeting! All my other friends and colleagues in the group, thank you for making everyday at the office amazing!

To my family, thank you for being there and supporting me every step of the way through my now 8+ years at Chalmers, and of course also before, through my whole life. Without your support and encouragement, I would not have come this far. Last but not least (well, probably you are the smallest!), thanks to Marléne for pushing me through this project (and a reminder just for you: $\arg(-j) = -\pi/2$)!

Christopher Lindberg
Gothenburg, October 2015

This research has been supported, in part, by the European Research Council under Grant No. 258418 (COOPNET).

Contents

Part I

Overview

Chapter 1

Introduction

1.1 Motivation

The advent of smart phones has made devices with wireless communication, computational, and sensing resources ubiquitous, providing a natural platform for the implementation of wireless sensor networks (WSNs). With the sharp increase in smart and communicating devices comes the opportunity to collect high-resolution sensing data, or collectively compute and perform advanced tasks. Due to the versatility of WSNs, they have been proposed for a myriad of applications including environmental monitoring [1, 2], weather monitoring [3], surveillance [4], tracking [5], and control and actuation [6]. However, the sheer size of these networks poses new challenges, and requires new solutions that tailor the network operations towards the specific applications they are used for. It is no longer possible or necessary to gather and process all data at a data fusion center, but the distributed nature of these systems should be utilized.

A family of processing algorithms operating in a distributed fashion that has gathered a significant amount of attention, is *consensus algorithms*. Consensus algorithms are iterative schemes where, in each iteration, sensors exchange information with their neighbors and make an update based on the rule specified by the algorithm with the aim of reaching a common value, i.e., consensus. The study of consensus problems can be traced back to the late 1950's statistical science and reaching consensus on opinions, where the works of [7–14] are relevant. Later, iterative algorithms to reach consensus for parallel computing and decision making were studied in [15, 16]. Consensus algorithms naturally found applications to cooperative coordination and control, building on the works of [17–22]. Some reasons why these algorithms are attractive for distributed processing in WSNs include: (i) The algorithm complexity scale well in the number of nodes, and almost no modifications are needed when sensors join or leave the network. (ii) They are robust to failure of sensors and communication links, and information delay and switching network topologies can be handled. (iii) The convergence properties can in many cases be analyzed using standard tools from linear algebra and graph theory. When implementing consensus algorithms to solve a distributed problem or task in a WSN, there are several considerations to take into account. Since many consensus algo-

rithms operate with asymptotic convergence behavior, i.e., they never converge to exact consensus, we should ask ourselves how large of an error in terms of disagreement is tolerable for our application. Additionally, since it is costly to perform many consensus iterations in terms of communication resources, this should be weighed against the desired error level. This thesis aims at investigating some of the trade-offs between time, communication resource, and disagreement that arise when using consensus algorithms to solve certain problems in WSNs.

1.2 Scope and Aim of the Thesis

In this thesis we study two types of trade-offs that arise from applying consensus algorithms in WSNs to understand how the system design affects the considered performance metrics. For the first trade-off we consider a network of sensors and a sink, where the average of the data from the WSN is desired at the sink. Hence, this information needs to be conveyed from the network to the sink. We derive a closed-form expression for the trade-off using upper bounds on the error of the consensus algorithms, which we then use to gain insight in how to design the system. The second trade-off takes into account the medium access control (MAC) of the communication network, and weighs the induced MAC delay against the averaging performance.

1.3 Organization of the Thesis

In Sweden, the Licentiate degree is an intermediate step for a doctoral student towards the final PhD degree, and this thesis serves as a documentation of the work towards this degree. This thesis is written as a collection of papers, and is divided into two parts, with Part I providing background information which serves to facilitate the reading of the appended papers in Part II. The remainder of Part I is organized as follows. Chapter 2 deals with the eigenvalues and eigenvectors of matrices associated to graphs. These concepts are then used to show how we can determine the convergence properties of certain consensus algorithms. In Chapter 3, we discuss two possible trade-offs that arise when implementing consensus algorithms in WSNs. Finally, in Chapter 4 we list the contributions of each of the appended papers.

Chapter 2

Distributed Consensus Algorithms

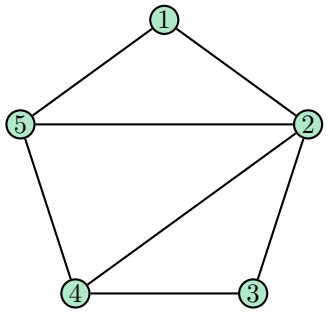
2.1 Introduction

The problem of distributed consensus is to make a network of nodes, or agents, agree on a value or state, with the nodes only having knowledge about and exchanging information with their neighbors. A distributed consensus algorithm is a set of instructions of how the nodes should exchange information, such that by iteratively performing the specified instructions, consensus is reached or approached. For example, we can think about a WSN where the nodes need to synchronize their clocks. This can be done by using a distributed consensus algorithm to reach consensus on the clock state [23]. There exists both continuous-time and discrete-time variations of consensus algorithms. Roughly, the continuous-time protocols describe the time-derivative of the states, whereas discrete-time protocols describe the state at consequent iterations. We are concerned with discrete-time distributed consensus algorithms, since the nature of WSNs pertain well to those protocols. In this chapter, we first introduce a basic graph-theoretic framework which helps us to analyze the convergence behavior of the aforementioned algorithms. Then, we mathematically describe the update rule of a distributed consensus algorithm, and analyze its convergence properties. Furthermore, we present a type of consensus protocols referred to as ratio consensus, where the consensus is on the ratio of two states.

2.2 Fundamentals

2.2.1 Basic Graph Theory

We consider a network of N sensors with capabilities to communicate with each other within some distance. To model how the nodes interact with each other, and how information moves in the network, we use a graph consisting of vertices, representing the sensors, and edges, representing the communication links between the sensors. The graph



$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Figure 2.1: An example of a labeled graph with five vertices connected over seven edges. Its adjacency matrix $\boldsymbol{A}$ and degree matrix $\boldsymbol{D}$ are shown to the right of the graph.

should be viewed as a tool to help us mathematically represent a network, such as a WSN, by abstracting the concept of nodes and communication links to mathematical sets. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is comprised of its vertex set $\mathcal{V}$ and its set of edges $\mathcal{E}$. Since the network we are considering consists of N nodes, we index them $i = 1, \ldots, N$. Consequently, the vertex set is given by $\mathcal{V} = \{1, 2, \ldots, N\}$, while the set of edges is equal to $\mathcal{E} = \{(i, j) \text{ if there is a link from } i \text{ to } j\}$. Note that for undirected graphs, which we are concerned with in this thesis, $(i, j) \in \mathcal{E}$ implies $(j, i) \in \mathcal{E}$. However generally for directed graphs this is not the case. For the undirected graph $\mathcal{G}$, we can also define the set of neighboring nodes of node i as $\mathcal{N}_i = \{j \in \mathcal{V} \,:\, (i, j) \in \mathcal{E}\}$. Hence, this set represents the nodes which node i can directly communicate with. The size of the set of neighbors of a node i is called its *degree*, denoted by $d_i = |\mathcal{N}_i|$. The maximum node-degree of the graph $\mathcal{G}$ is denoted by $d_{\max} = \max_i d_i$. To avoid working only with indexed sets we define the following useful matrices

- The *adjacency matrix* $\boldsymbol{A}$, which is a mapping of the set of edges $\mathcal{E}$ into a binary $N \times N$ matrix. The elements of the adjacency matrix $\boldsymbol{A}$ are defined as $A_{ij} = \mathbb{I}_{\mathcal{E}}(i, j)$, where $\mathbb{I}_{\mathcal{E}}(i, j) = 1$ if $(i, j) \in \mathcal{E}$.

- The *degree matrix* $\boldsymbol{D}$, defined as $\boldsymbol{D} = \text{diag}(\boldsymbol{A}\mathbf{1})$, where $\mathbf{1}$ is the all-one vector of appropriate size. The diagonal element D_{ii} is equal to the degree of node i.

- The graph *Laplacian matrix* $\boldsymbol{L}$, defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$.

In Figure 2.2.1 we show an example of what the adjacency and degree matrices would look like for a specific graph. The eigenvalues and eigenvectors of these matrices can in many cases be used to determine convergence properties, such as bounding the convergence rate, and show which value a consensus algorithm converges to.

2.2.2 Eigenvalues and Eigenvectors

An *eigenvector* $\boldsymbol{v}$ of a square matrix $\boldsymbol{M}$ is a column vector that when multiplied with $\boldsymbol{M}$ is scaled by the *eigenvalue* λ, i.e.,

$$\boldsymbol{M}\boldsymbol{v} = \lambda\boldsymbol{v}. \tag{2.1}$$

Similarly, a *left eigenvector* $\boldsymbol{u}^\mathsf{T}$ of the matrix $\boldsymbol{M}$ is a row vector such that

$$\boldsymbol{u}^\mathsf{T}\boldsymbol{M} = \lambda\boldsymbol{u}^\mathsf{T}. \tag{2.2}$$

The eigenvalues of a matrix $\boldsymbol{M}$ can be found by solving the characteristic polynomial $\det(\boldsymbol{M} - \lambda\boldsymbol{I}) = 0$. The following definition is useful in order to understand how eigenvalues and eigenvectors can be used to decompose a matrix.

Definition 1 *A matrix $\boldsymbol{M} \in \mathbb{R}^{N\times N}$ is diagonalizable if there exists a diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{N\times N}$, such that*

$$\boldsymbol{M}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}, \tag{2.3}$$

for some matrix $\boldsymbol{V} \in \mathbb{R}^{N\times N}$.

If we let the columns of $\boldsymbol{V}$ be the eigenvectors of $\boldsymbol{M}$, we see immediately from (2.3) that the diagonal entries of $\boldsymbol{\Lambda}$ must be the corresponding eigenvalues of those eigenvectors (cf. (2.1)). This is referred to as eigendecomposition of the matrix $\boldsymbol{M}$, and is usually written as $\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$. The matrix $\boldsymbol{V}$ is usually referred to as the eigenbasis of $\boldsymbol{M}$. We also note, since (2.3) can be written as $\boldsymbol{V}^{-1}\boldsymbol{M} = \boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, that $\boldsymbol{U} \triangleq \boldsymbol{V}^{-1}$, where the rows of $\boldsymbol{U}$ are the left eigenvectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ of $\boldsymbol{M}$. Using eigendecomposition of $\boldsymbol{M}$, we have

$$\boldsymbol{M}^2 = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} \tag{2.4}$$

$$= \boldsymbol{V}\boldsymbol{\Lambda}^2\boldsymbol{V}^{-1}, \tag{2.5}$$

and if we iteratively apply the same method, we conclude that

$$\boldsymbol{M}^k = \boldsymbol{V}\boldsymbol{\Lambda}^k\boldsymbol{V}^{-1}, \tag{2.6}$$

or alternatively, that $\boldsymbol{M}^k\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}^k$. Now, if we multiply with an arbitrary vector $\boldsymbol{\alpha}$ from the right on both sides of that expression, we get

$$\boldsymbol{M}^k\boldsymbol{V}\boldsymbol{\alpha} = \boldsymbol{V}\boldsymbol{\Lambda}^k\boldsymbol{\alpha} \tag{2.7}$$

$$= \sum_{i=1}^{N} \boldsymbol{v}_i\lambda_i^k\alpha_i, \tag{2.8}$$

where the $\boldsymbol{v}_i$ is the ith eigenvector of $\boldsymbol{M}$ with corresponding eigenvalue λ_i's, and α_i the ith element of the vector $\boldsymbol{\alpha}$.

2.3 Consensus Problem

Let each node $i = 1, \ldots, N$ have an initial value $x_i(0)$, and stack the values of all nodes in a vector $\boldsymbol{x}(0) = [x_1(0), \ldots, x_N(0)]^\mathsf{T}$. Also, let $\boldsymbol{x}(k) = [x_1(k), \ldots, x_N(k)]^\mathsf{T}$ denote the nodes' values after k iterations of information exchange and updating according to a consensus algorithm. By consensus problem we mean the problem of finding an update rule such that as the nodes exchange information with their neighbors according to this rule, they all end up with the same value. Especially, as we let the number of information exchange rounds go to infinity, we would like to have

$$\lim_{k\to\infty} \boldsymbol{x}(k) = c\mathbf{1}, \tag{2.9}$$

where $c \in \mathbb{C}$ is some constant. In other words, such that the nodes reach agreement (consensus) on a value c. A special case of this problem is the average consensus problem. In that case, the constant c is equal to the average, i.e.,

$$c = \frac{1}{N}\sum_{i=1}^{N} x_i(0) \triangleq \bar{x}. \tag{2.10}$$

In the next section we present some examples of average consensus algorithms, and use the spectral properties of $\mathcal{G}$ to show that these algorithms indeed converge to the average in all nodes. Usually, the network dynamics of a consensus algorithm is described as a matrix power series, such that the state at iteration k is

$$\boldsymbol{x}(k) = \boldsymbol{P}\boldsymbol{x}(k-1), \tag{2.11}$$

where $P_{ij} = 0$ when $(i,j) \notin \mathcal{E}$. The elements of $\boldsymbol{P}$, which are defined through the algorithm updating equations, decide the convergence properties of this power series. More specifically, how fast the algorithm converges through the eigenvalues of $\boldsymbol{P}$, and the consensus value c through the left eigenvectors of $\boldsymbol{P}$.

2.4 Consensus Algorithms

2.4.1 Uniform Weight Consensus

One of the more common distributed average consensus algorithms (or distributed averaging algorithms), is one first proposed in [21], which we will refer to as uniform weight consensus. The nodes update their value at each iteration according to the following expression

$$x_i(k) = x_i(k-1) + \xi \sum_{j\in\mathcal{N}_i} \left(x_j(k-1) - x_i(k-1)\right), \tag{2.12}$$

where $0 < \xi < d_{\max}$ is the algorithm step-size, agreed upon by all nodes before initiating the consensus algorithm[1]. The step-size can be agreed upon by using for example max-consensus algorithm on the node degree, which converges in finite time. The update rule

[1] Choosing a uniform step-size for all nodes is suboptimal in terms of convergence rate. For more sophisticated step-size selection, see the so called Metropolis weights in [24].

of a finite-time max-consensus algorithm is

$$x_i(k) = \max_{j \in \mathcal{N}_i \cup \{i\}} x_j(k-1), \tag{2.13}$$

i.e., all nodes update to the maximum value in their local neighborhood at iteration $k-1$. In the case of running max-consensus on the node degree, the initial values would be set to $x_i(0) = d_i$. Now, for the uniform weight consensus, if we sum over all nodes at iteration k, we get

$$\sum_{i=1}^{N} x_i(k) = \sum_{i=1}^{N} x_i(k-1) + \sum_{i=1}^{N} \xi \sum_{j \in \mathcal{N}_i} \left(x_j(k-1) - x_i(k-1) \right). \tag{2.14}$$

The term involving a double-sum is equal to zero since the graph $\mathcal{G}$ is undirected. Hence, summing over all values at iteration k is equal to the sum of the values at iteration $k-1$. By recursion, we deduce that

$$\sum_{i=1}^{N} x_i(k) = \sum_{i=1}^{N} x_i(0). \tag{2.15}$$

Hence, the algorithm is sum (or average) preserving. Consequently, if the algorithm converges to a consensus value, it must be the average by this property. By looking at (2.12) and using the graph representing matrices defined in Section 2.2.1, the algorithm dynamics of the whole network can be written as

$$\boldsymbol{x}(k) = \boldsymbol{P}\boldsymbol{x}(k-1), \tag{2.16}$$

where $\boldsymbol{P} = \boldsymbol{I} - \xi \boldsymbol{L}$. By recursion, it also holds that

$$\boldsymbol{x}(k) = \boldsymbol{P}^k \boldsymbol{x}(0). \tag{2.17}$$

Comparing these two expressions to the eigendecomposition and related expressions, especially (2.8), we see that expressing $\boldsymbol{x}(0)$ in the coordinates of the eigenbasis of $\boldsymbol{P}$, i.e., let $\boldsymbol{\alpha} = \boldsymbol{V}^{-1}\boldsymbol{x}$, we can rewrite the right hand side of (2.17) as

$$\boldsymbol{P}^k \boldsymbol{V} \boldsymbol{\alpha} = \sum_{i=1}^{N} \lambda_i^k \alpha_i \boldsymbol{v}_i, \tag{2.18}$$

where the columns of $\boldsymbol{V}$ are the eigenvectors of $\boldsymbol{P}$, λ_i's are the eigenvalues, and α_i are the coordinates of $\boldsymbol{x}(0)$ in the eigenbasis of $\boldsymbol{P}$. Now, by inspecting (2.18), we see that in order for this power series to converge, we need the eigenvalues of $\boldsymbol{P}$ to be inside the unit circle, i.e., $|\lambda_i| \leq 1$ for all i. We trivially find that $\boldsymbol{v}_1 = \boldsymbol{1}$ is an eigenvector of $\boldsymbol{P}$ with corresponding eigenvalue $\lambda_1 = 1$ by

$$\boldsymbol{P}\boldsymbol{1} = (\boldsymbol{I} - \xi \boldsymbol{L})\boldsymbol{1} \tag{2.19}$$

$$= \boldsymbol{1}, \tag{2.20}$$

since $\mathbf{1}$ is in the null space of the Laplacian matrix $\boldsymbol{L}$. Since $\lambda_1^k = 1$ the part of $\boldsymbol{x}(0)$ aligned in the direction of $\boldsymbol{v}_1 = \mathbf{1}$ is the part that stays constant through the consensus iterations. Consequently, α_1 is the consensus value, hence $\alpha_1 = \bar{x}$. The components of $\boldsymbol{x}(0)$ aligned in the directions of $\boldsymbol{v}_2, \ldots, \boldsymbol{v}_N$, on the other hand, are considered as disagreement. By (2.18), we have

$$\boldsymbol{P}^k \boldsymbol{x}(0) = \lambda_1^k \alpha_1 \boldsymbol{v}_1 + \sum_{i=2}^{N} \lambda_i^k \alpha_i \boldsymbol{v}_i \tag{2.21}$$

$$= \bar{x}\mathbf{1} + \delta(k), \tag{2.22}$$

where $\delta(k)$ is the disagreement at iteration k. In [25] it is proven that $\lambda_1 = 1$ is an eigenvalue of multiplicity 1, and that the eigenvalues of $\boldsymbol{P}$ are inside the unit circle. This is done by using the fact that $\boldsymbol{P}$ is a stochastic and primitive matrix, and applying the Perron-Frobenius theorem for nonnegative matrices. Hence, we can order the eigenvalues of $\boldsymbol{P}$ as $1 = \lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_N|$. The disagreement at iteration k can be upper bounded as

$$\delta(k) = \sum_{i=2}^{N} \lambda_i^k \alpha_i \boldsymbol{v}_i \tag{2.23}$$

$$\leq |\lambda_2^k| \delta(0), \tag{2.24}$$

and since $\delta(k) = \boldsymbol{x}(k) - \bar{x}\mathbf{1}$, we have that the ℓ_2-norm at iteration k is upper bounded by

$$\|\boldsymbol{x}(k) - \bar{x}\mathbf{1}\|_2 \leq |\lambda_2^k| \|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2. \tag{2.25}$$

This means that the disagreement among the sensors, or the consensus error, is monotonically decreasing in the number of iterations k, and decreases at least as fast as stated in (2.25). Furthermore, the asymptotic convergence rate is exactly equal to $|\lambda_2^k|$.

2.4.2 Ratio Consensus

Applying the consensus algorithm presented in the previous section requires all nodes to be associated with a single and unique value in order to guarantee convergence to the average. If indeed there exist several copies of values, and if some nodes are associated with several or no values, we can use an algorithm commonly referred to as *ratio consensus*. Ratio consensus computes the average in a distributed fashion by using two consensus processes running in parallel. The processes may not converge independently to the average, but the ratio of the two does. It should be pointed out that the ratio consensus algorithm is not only used in the case when data association is not unique, but can be used in the normal case as well. The algorithm, though in a weighted gossip setting, is first proposed in [26], however, as pointed out in [27], the background and idea of the ratio of such algorithms can be traced back earlier (see Bibliography and Discussion to 3.1-3.2 in [28]). The two consensus processes iterate over the data $\boldsymbol{x} = [x_1, \ldots, x_N]^\mathsf{T}$ and the so called ratio consensus weights $\boldsymbol{w} = [w_1, \ldots, w_N]^\mathsf{T}$, respectively. The update rule

for the two processes is

$$x_i(k) = \frac{1}{d_i + 1} x_i(k-1) + \sum_{j \in \mathcal{N}_i} \frac{1}{d_j + 1} x_j(k-1), \tag{2.26}$$

and

$$w_i(k) = \frac{1}{d_i + 1} w_i(k-1) + \sum_{j \in \mathcal{N}_i} \frac{1}{d_j + 1} w_j(k-1). \tag{2.27}$$

The update dynamics in matrix form can be expressed as

$$\boldsymbol{x}(k) = \boldsymbol{P}\boldsymbol{x}(k-1), \tag{2.28}$$

where $\boldsymbol{P} = (\boldsymbol{D} + \boldsymbol{I})^{-1}(\boldsymbol{A} + \boldsymbol{I})$. It is readily verified that the consensus matrix $\boldsymbol{P}$ is a column stochastic matrix, but not necessarily row stochastic (the special case of a doubly stochastic $\boldsymbol{P}$ appears when $\mathcal{G}$ is an n-regular graph with $n+1$ nodes). Hence, in general $\boldsymbol{P}$ does not have an eigenvector $\boldsymbol{v} = \mathbf{1}$ with corresponding eigenvalue $\lambda = 1$. Consequently, each consensus process does not in general converge to a consensus value, but to a stationary distribution which we denote by $\boldsymbol{\pi}$. Here, we present two scenarios, one with unique and one with nonunique data association, for which we describe the initialization procedure and show what the algorithm asymptotically converges to.

2.4.2.1 Scenario 1

When each node in the network is associated with a unique value, the consensus algorithms are initialized with the two vectors $\boldsymbol{x} = [x_1, \ldots, x_N]^\mathsf{T}$ and $\boldsymbol{w} = \mathbf{1}$, respectively. The result is then calculated by each node taking the ratio of the two values, i.e.,

$$\hat{\boldsymbol{x}}(k) = \boldsymbol{P}^k \boldsymbol{x}(0) \oslash \boldsymbol{P}^k \boldsymbol{w}(0), \tag{2.29}$$

where $\hat{\boldsymbol{x}}(k)$ is the vector of average estimates at iteration k, and $\oslash$ denotes the Hadamard division, i.e., element-wise division of two matrices or vectors of similar dimensions. Since each column of $\boldsymbol{P}$ converges to the stationary distribution $\boldsymbol{\pi}$, the limit of the ratio vector $\hat{\boldsymbol{x}}$ is

$$\lim_{k\to\infty} \hat{\boldsymbol{x}}(k) = \lim_{k\to\infty} \boldsymbol{P}^k \boldsymbol{x}(0) \oslash \lim_{k\to\infty} \boldsymbol{P}^k \boldsymbol{w}(0) \tag{2.30}$$

$$= \sum_{i=1}^{N} \boldsymbol{\pi} x_i(0) \oslash \sum_{i=1}^{N} \boldsymbol{\pi} w_i(0) \tag{2.31}$$

$$= \frac{\sum_{i=1}^{N} x_i(0)}{\sum_{i=1}^{N} w_i(0)} (\boldsymbol{\pi} \oslash \boldsymbol{\pi}) \tag{2.32}$$

$$= \frac{\sum_{i=1}^{N} x_i(0)}{N} \mathbf{1} \tag{2.33}$$

$$= \bar{x}\mathbf{1}. \tag{2.34}$$

Thus, the ratio consensus algorithm converges to the average of the initial values of the nodes for unique data association.

2.4.2.2 Scenario 2

For the scenario where we have non-unique data and where each node might be associated with several values, the initialization of the consensus algorithms is done as follows: Let β_j denote the jth unique value present in the network, and let $\mathcal{B}_j$ be the set of nodes that are associated with β_j. Then, the initial values of node i are

$$x_i(0) = \sum_{j:i\in\mathcal{B}_j} \frac{\beta_j}{|\mathcal{B}_j|} \tag{2.35}$$

$$w_i(0) = \sum_{j:i\in\mathcal{B}_j} \frac{1}{|\mathcal{B}_j|}. \tag{2.36}$$

With these modified initial values, we can do the same exercise as for the unique data scenario, but now our goal is to compute $\bar{\beta} \triangleq (1/N_\beta)\sum_{j=1}^{N_\beta} \beta_j$, where N_β is the number of unique values in the network. Asymptotically in the number of iterations k, we have

$$\lim_{k\to\infty} \hat{\boldsymbol{x}}(k) = \lim_{k\to\infty} \boldsymbol{P}^k \boldsymbol{x}(0) \oslash \lim_{k\to\infty} \boldsymbol{P}^k \boldsymbol{w}(0) \tag{2.37}$$

$$= \sum_{i=1}^{N} \boldsymbol{\pi} x_i(0) \oslash \sum_{i=1}^{N} \boldsymbol{\pi} w_i(0) \tag{2.38}$$

$$= \frac{\sum_{i=1}^{N} x_i(0)}{\sum_{i=1}^{N} w_i(0)} (\boldsymbol{\pi} \oslash \boldsymbol{\pi}) \tag{2.39}$$

$$= \frac{\sum_{j=1}^{N_\beta} \beta_j}{N_\beta} \mathbf{1} \tag{2.40}$$

$$= \bar{\beta}\mathbf{1}. \tag{2.41}$$

Hence, with the initialization described by (2.35) and (2.36), we asymptotically achieve average consensus of the unique β_j's. To derive the convergence rate of the ratio consensus algorithm, we cannot proceed as for the uniform weight consensus algorithm, where we used the Perron-Frobenius theorem. The main reason for this is that the algorithm includes an element-wise division operation. The convergence rate of ratio consensus is investigated in detail in [29], and in Paper C we derive an upper bound.

Chapter 3

Trade-offs in Distributed Consensus for Wireless Sensor Networks

In this section, we present two trade-offs which arise when implementing a consensus algorithm in a WSN. With trade-off we mean that there is not only one single set of parameters that would solve the desired problem, but many different values of the adjustable system parameters could do, and changing one value affects the other parameters of the system. Hence, these parameters should be traded off against each other, and different applications will lead to different preferred combinations depending on the requirements the specific application entails.

3.1 Local vs. Global Communication Cost

As for the standard consensus problem, let the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph representing the WSN. However, now we also consider a sink, or fusion center, that is not part of $\mathcal{G}$, but which desires to know $\bar{x}$ (see Figure 3.1 for an illustration of the system). The sink is assumed to know the number of nodes in the network, i.e., $N = |\mathcal{V}|$. Furthermore, we consider every node to be able to communicate with the sink, where each transmission incurs a cost C_{glo}. In contrast, every transmission over the sensor-to-sensor channel is associated with a cost C_{loc}. There are two ways that we immediately realize achieve the goal of obtaining $\bar{x}$ at the sink:

- Every sensor sends its associated value directly to the sink over the sensor-to-sink communication channel. The sink then sums the received data and divides by the number of nodes in the WSN.
- An infinite number of consensus iterations are run in the network using sensor-to-sensor communication to perform consensus updates. After this, any sensor in the network is chosen to send the average to the sink.

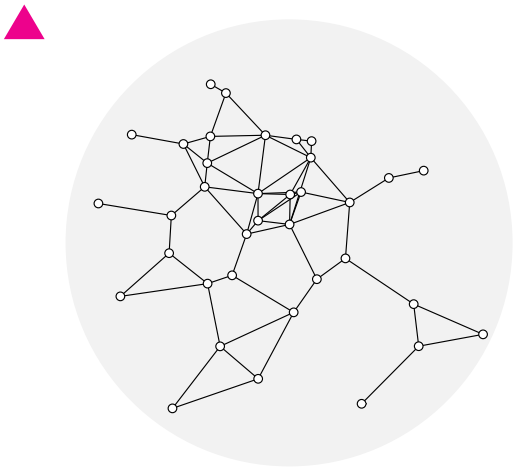


Figure 3.1: An illustration of the system considered for the trade-off on local vs. global communication cost, showing the sink as a magenta-colored triangle, and the wireless sensor network with the local communication graph.

If we limit ourselves to conveying the exact average to the sink, these are the options we are faced with. Since we cannot wait an infinite amount of time, we choose the first solution. However, things get more interesting if we allow for a certain level of error to the average, i.e., if the sink is satisfied knowing $\bar{x} \pm \varepsilon$. Since we allow for some error, we can let the number of sensors that we query be less than the total number of sensors, i.e., $L \leq N$, and we can also have $I < \infty$. In order to understand how these two parameters affect the trade-off between local and global communication, we characterize the error as a function of k and L. Assume that the sink uses simple random querying of nodes, i.e., each node is selected with equal probability without replacement, and that $\bar{x}$ is estimated with the unbiased mean estimator

$$\hat{x}(k) = \frac{1}{L} \sum_{i \in \mathcal{L}} x_i(k), \tag{3.1}$$

where $\mathcal{L}$ is the set of queried nodes, with $|\mathcal{L}| = L$. Then, the estimation at the sink, after k iterations of uniform weight consensus, can be decomposed as

$$\hat{x}(k) = \frac{1}{L} \sum_{i \in \mathcal{L}} x_i(k) \tag{3.2}$$

$$= \bar{x} + \frac{1}{L} \sum_{i \in \mathcal{L}} \Delta_i(k) \tag{3.3}$$

$$= \bar{x} + e, \tag{3.4}$$

where e is a random variable denoting the estimation error due to consensus and the estimation of $\bar{x}(k)$ at the sink. Since the consensus algorithm is mean preserving, as mentioned in Section 2.4.1, it is easy to see that the average of the estimation error at the sink is $\mathbb{E}[e] = 0$, where the average is over the queried set $\mathcal{L}$. Due to using simple random querying of the sensors, considering many realizations of querying, the variance of e is [30]

$$\mathrm{Var}(e) = \frac{1}{L}\left(\frac{N-L}{N-1}\right)\sigma_x^2(k), \tag{3.5}$$

in which $\sigma_x^2(k)$ is the population variance among the sensors in the WSN after k iterations. By definition, this variance is calculated by $\sigma_x^2(k) = \|\boldsymbol{x}(k) - \bar{x}\mathbf{1}\|_2^2$, and since we use a uniform weight consensus algorithm the upper bound on the consensus error using the spectral properties of $\mathcal{G}$ gives

$$\sigma_x^2(k) \leq \lambda_2^{2k}\|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2^2, \tag{3.6}$$

where λ_2 is the second largest eigenvalue by magnitude of the consensus matrix. Consequently, we can upper bound the variance of the estimation error by

$$\mathrm{Var}(e) \leq \lambda_2^{2k}\|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2^2\frac{1}{L}\left(\frac{N-L}{N-1}\right). \tag{3.7}$$

This leads us to the following proposition.

Proposition 1 *Assuming that $|\lambda_2| < 1$, the minimum number of consensus iterations k^* as a function of the number of queried sensors L needed to guarantee $\sigma_e \leq \varepsilon$ is given by*

$$k^*(L) = \left\lceil \left(\log\varepsilon - \log\|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2 - \frac{1}{2}\log\frac{1}{L}\left(\frac{N-L}{N-1}\right)\right) / \log|\lambda_2| \right\rceil^+, \tag{3.8}$$

where $(\cdot)^+ = \max(\cdot, 0)$.

We observe that querying all N sensors, i.e., setting $L = N$ yields

$$-\frac{1}{2}\log\frac{1}{L}\left(\frac{N-L}{N-1}\right) = -\frac{1}{2}\log 0 \tag{3.9}$$

$$= +\infty. \tag{3.10}$$

Since we assume that the consensus algorithm converges, we know that $|\lambda_2| < 1$. So, the expression for $k^*(N)$ is

$$k^*(N) \propto -\left\lceil \log\varepsilon - \log\|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2 - \frac{1}{2}\log 0 \right\rceil^+ \tag{3.11}$$

$$= \left(\log\|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2 - \log\varepsilon + \frac{1}{2}\log 0\right)^+ \tag{3.12}$$

$$= 0. \tag{3.13}$$

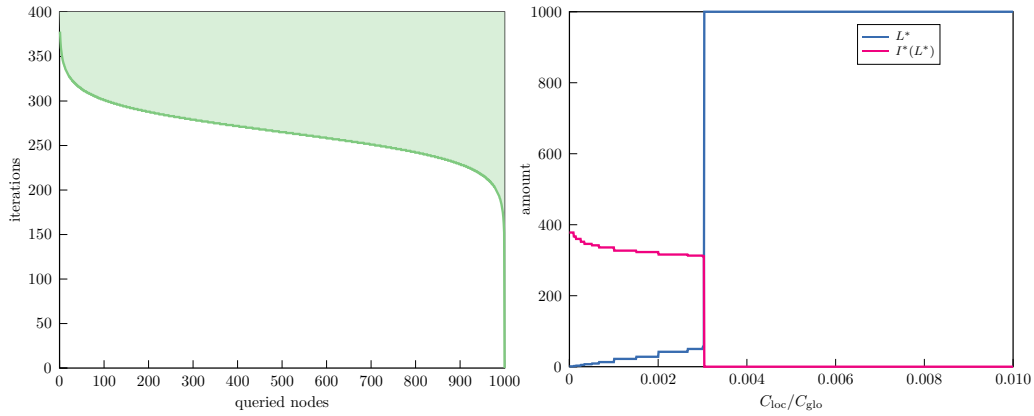


Figure 3.2: The figure to the left shows the region of points satisfying a given error threshold ε for a given system (note that $N = 1000$). Any combination of iterations and number of queried nodes within the green area would satisfy the error threshold condition for the specific system. To the right we show for the same system the cost-optimal combination of iterations and queried nodes given a certain communication cost ratio $C_{\text{loc}}/C_{\text{glo}}$.

It makes sense that if we choose $L = N$, then no consensus iterations are needed. If we look at the other end of the spectrum, i.e., setting $L = 1$, we end up with the following expression

$$k^*(1) = \left(\left(\log \varepsilon - \log \|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2 - \frac{1}{2} \log 1 \right) / \log |\lambda_2| \right)^+ \tag{3.14}$$

$$= \lceil (\log \varepsilon - \log \|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2) / \log |\lambda_2| \rceil^+ . \tag{3.15}$$

This equation only consists of the initial consensus disagreement $\|\boldsymbol{x}(0) - \bar{x}\mathbf{1}\|_2$, the error threshold ε, and λ_2. Hence, the error is independent of the estimation at the sink. Proposition 1 enables us to choose how to operate our system with a specified error tolerance ε. The cost of operating a specific system is calculated as

$$C_{\text{tot}} = C_{\text{glo}} L + C_{\text{loc}} N k^*(L). \tag{3.16}$$

In order to find the cost optimal system parameters we thus want to determine

$$L^* = \underset{1 \le L \le N}{\arg\min} \left(L + \frac{C_{\text{loc}}}{C_{\text{glo}}} N k^*(L) \right), \tag{3.17}$$

for a given local-global cost ratio $C_{\text{loc}}/C_{\text{glo}}$. The local-global cost ratio is application specific, but also depends on which cost is considered, e.g., energy consumption or bandwidth usage. The trade-off is visualized in Figure 3.1, where we show the number of iterations k^* that would satisfy the error threshold ε for a given number of queried nodes

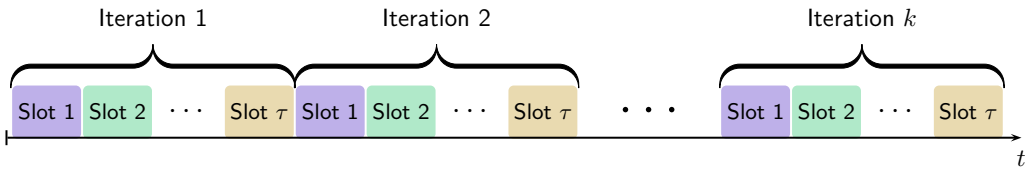


Figure 3.3: The total delay of performing k iterations of consensus, with τ being the length of the S-TDMA protocol. In each slot, a portion of the nodes broadcast their information, and these nodes are spatially separated at a distance such that they do not cause any packet collisions.

L. The system parameters used to produce the results include $N = 1000$, $\varepsilon = 0.01$, and $\lambda_2 = 0.95$. We observe that when the number of queried nodes is between $0.1N$ and $0.9N$ the curve is rather flat. This means that instead of querying 90% of the nodes, we could query only 10% and perform a few more consensus iterations, which would probably save us a large cost. Of course, this depends on the communication cost ratio. Hence, in the right plot of Figure 3.1, we also numerically evaluate the communication cost optimal system for a given communication cost ratio $C_{\text{loc}}/C_{\text{glo}}$. Interestingly, we see a strong threshold effect in what kind of estimation (consensus or sink) that should be used. In more detail this means that if $C_{\text{loc}}/C_{\text{glo}}$ is low we should go for almost only consensus iterations and sensor-to-sensor communication. On the other hand, if $C_{\text{loc}}/C_{\text{glo}}$ is high, there is no point in performing consensus, since it is too expensive, and we can let every node transmit to the sink instead.

3.2 Delay vs. Performance

For the cost vs. performance trade-off, we consider a WSN of N sensors where each sensor desires to know the average of the values observed by the sensors. The observation of sensor i is denoted by x_i and all the values are stacked in the vector $\boldsymbol{x} = [x_1, \ldots, x_N]^{\mathsf{T}}$. The information exchange between sensors is performed over a shared channel using broadcast, with connectivity modeled according to a disc model with radius R. To avoid packet collisions, broadcasts are scheduled using time division multiple access (TDMA) with spatial reuse. The TDMA protocol assigns a time slot to each node within which it is allowed to broadcast its information. With spatial reuse, nodes separated by a distance greater than R may be scheduled in the same time slot. An iteration of the consensus algorithm is completed once every node has broadcasted its information. Thus, we consider every iteration of consensus to incur a delay $\tau(R)$. This is illustrated in Figure 3.1. The delay $\tau(R)$ is the length of the TDMA schedule[1], and the length of this depends on the communication radius R through the spatial reuse of the medium access control (MAC) protocol as a monotonically increasing function of R. Moreover, the convergence rate (i.e., performance) of the consensus algorithm is a monotonically

[1]It is computationally complex to find the exact value of τ for a certain network. Hence, we resort to the lower bound $\tau \geq 1 + d_{\max}$.

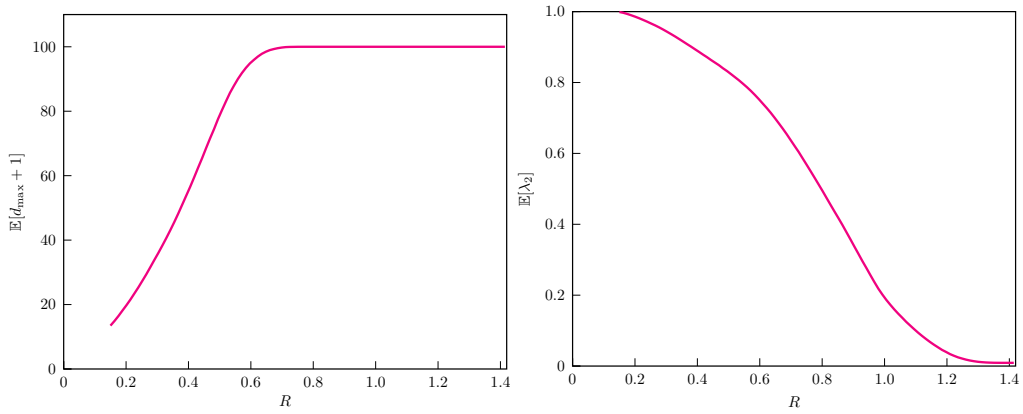


Figure 3.4: To the left, we show the mean value of the lower bound of τ for a given radius R. To the right, we show the mean value of λ_2. The WSN was simulated with $N = 100$ over a $1\,\mathrm{m} \times 1\,\mathrm{m}$ area.

increasing function in R as well, since $|\lambda_2|$ decreases when R increases (see Figure 3.4 for visualization of the two values discussed). Hence, if we have the possibility to freely choose R, it is important to study the inherent trade-off between the resulting MAC delay and performance in order to tailor the system to the specific application it is applied. We address the following questions: do we need a fast or exact system, and how do we achieve it?

Chapter 4

Contributions

4.1 Included Publications

1. **Paper A: "MAC Delay in Belief Consensus for Distributed Tracking"**
   In this paper, we study the trade-off between MAC delay and performance for distributed particle filtering using uniform weight consensus in a WSN, where the particle filter is used in a distributed tracking application. We impose an S-TDMA MAC protocol to schedule the broadcasts used to run consensus on the particles of the Bayesian filter. The choice of communication radius affects both delay and performance of the algorithm, e.g., with high connectivity comes better performance per iteration, but each iteration takes longer time to execute. We found that for small networks, (i) the impact of the max-consensus part of the tracking algorithm should be accounted for; (ii) a simple round-robin schedule combined with a large communication range gives the best delay/performance trade-off.

2. **Paper B: "Distributed Compressed Sensing for Sensor Networks with Packet Erasures"**
   We study two approaches to distributed compressed sensing for in-network data compression and signal reconstruction at a sink, where the sensors are equispaced on a line. Of the two approaches, one uses clustering of the nodes and the other uses uniform weight consensus to compress the data with random linear projections. In this paper, the local communication cost tied to the in-network compression is considered to be delay due to using S-TDMA to schedule the sensors' broadcasts. Sensor-to-sink communication on the other hand, is associated with a bandwidth cost. Hence, we investigate the trade-off between bandwidth cost and delay given a certain reconstruction performance requirement when using basis pursuit denoising (ℓ_1-minimization) for reconstruction. Moreover, we analyze and compare the performance degradation due to erased packets sent to the sink.

3. **Paper C: "Compressed Sensing for Wireless Sensor Networks Without Explicit Position Information"**
   In this paper, we propose a framework for in-network compression without the sink

knowing the positions of all the sensors, using compressed sensing and spatial interpolation. Reconstruction in compressed sensing relies on knowledge of a sparsifying transform, which in the setting of a WSN would require position information from all sensors transmitting to the sink. We circumvent this by only using the position information locally performing spatial interpolation at known locations. The interpolation step is followed by a distributed compression step implemented by random linear projections and ratio consensus using local communication between sensors. Reconstruction of the data is performed in three steps: (i) After ratio consensus is finished, query a subset of the nodes to send their compression estimates, using sensor-to-sink communication. (ii) Combine the received estimates by taking the average. (iii) Solve the underdetermined system that is the decompression by ℓ_1-minimization. We investigate the trade-off arising from choosing the number of ratio consensus iterations and the number of nodes to query, given an error tolerance.

References

[1] K. Martinez, J. K. Hart, and R. Ong, "Environmental sensor networks," *Computer*, vol. 37, no. 8, pp. 50–56, Aug. 2004.

[2] L. M. Oliveira and J. J. Rodrigues, "Wireless sensor networks: a survey on environmental monitoring," *Journal of Communications*, vol. 6, no. 2, pp. 143–151, Apr. 2011.

[3] C. Yawut and S. Kilaso, "A wireless sensor network for weather and disaster alarm systems," in *Proc. Int. Conf. on Information and Electronics Engineering*, vol. 6, May 2011, pp. 155–159.

[4] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-efficient surveillance system using wireless sensor networks," in *Proceedings of the 2nd Int. Conf. on Mobile systems, applications, and services*, Jun. 2004, pp. 270–283.

[5] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: a wireless sensor network for target detection, classification, and tracking," *Computer Networks*, vol. 46, no. 5, pp. 605–634, Dec. 2004.

[6] F. Xia, Y.-C. Tian, Y. Li, and Y. Sung, "Wireless sensor/actuator network design for mobile control applications," *Sensors*, vol. 7, no. 10, pp. 2157–2173, Oct. 2007.

[7] E. Eisenberg and D. Gale, "Consensus of subjective probabilities: The pari-mutuel method," *The Annals of Mathematical Statistics*, vol. 30, no. 1, pp. 165–168, Mar. 1959.

[8] M. Stone, "The opinion pool," *The Annals of Mathematical Statistics*, vol. 32, no. 4, pp. 1339–1342, Dec. 1961.

[9] A. Madansky, "Externally bayesian groups," RAND Corporation, Tech. Rep. RM-4141-PR, 1964.

[10] T. Norvig, "Consensus of subjective probabilities: A convergence theorem," *The Annals of Mathematical Statistics*, vol. 38, no. 1, pp. 221–225, Feb. 1967.

[11] R. L. Winkler, "The consensus of subjective probability distributions," *Management Science*, vol. 15, no. 2, pp. B–61, Oct. 1968.

[12] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, Mar. 1974.

[13] S. Chatterjee and E. Seneta, "Towards consensus: some convergence theorems on repeated averaging," *Journal of Applied Probability*, vol. 14, no. 1, pp. 89–97, Mar. 1977.

[14] G. L. Gilardoni and M. K. Clayton, "On reaching a consensus using degroot's iterative pooling," *The Annals of Statistics*, vol. 21, no. 1, pp. 391–401, Mar. 1993.

[15] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[16] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

[17] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[18] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the American Controls Conference*, vol. 2, Jun. 2003, pp. 951–956.

[19] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 622–629, Apr. 2004.

[20] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.

[21] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[22] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[23] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Proceedings of the 46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 2289–2294.

[24] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proceedings of the 4th Int. Symp. on Information Processing in Sensor Networks*, Apr. 2005, pp. 63–70.

[25] R. Olfati-Saber, A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[26] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Ann. IEEE Symp. on Foundations of Computer Science*, Oct. 2003, pp. 482–491.

[27] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, Mar. 2014.

[28] E. Seneta, *Non-negative matrices and Markov chains*. Springer Science & Business Media, 2006.

[29] F. Iutzeler, "Distributed estimation and optimization for asynchronous networks," Ph.D. dissertation, Telecom ParisTech, 2013.

[30] J. Rice, *Mathematical statistics and data analysis*. Cengage Learning, 2006.

Part II

Included papers

Paper A

MAC Delay in Belief Consensus for Distributed Tracking

Christopher Lindberg, L. Srikar Muppirisetty, Karl-Magnus Dahlén, Vladimir Savic, and Henk Wymeersch

Published in
Proceedings of the 10th Workshop on Positioning, Navigation and Communication,
Dresden, Germany, Apr. 2013

Paper B

Distributed Compressed Sensing in Sensor Networks with Packet Erasures

Christopher Lindberg, Alexandre Graell i Amat, and Henk Wymeersch

Published in
Proceedings of the IEEE Global Conference on Communication (GLOBECOM)
Austin, TX, USA, Apr. 2014

Paper C

Compressed Sensing in Wireless Sensor Networks Without Explicit Position Information

Christopher Lindberg, Alexandre Graell i Amat, and Henk Wymeersch

In preparation.