Efficient Saptio-Temporal Sampling in Wireless Sensor Networks Based on Compressive Sampling

Vom Fachbereich Informatik der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines Doktor-Ingenieur (Dr.-Ing.)
vorgelegt von

M.Sc. Mohammadreza Mahmudimanesh

aus Zabol, Iran

Referenten:
Prof. Neeraj Suri, Ph.D.
Prof. Jiannong Cao

Datum der Einreichung: 21. Mai 2015
Datum der mündlichen Prüfung: 7. Juli 2015

Darmstadt 2015
D17

Abstract

A wireless sensor network monitors the environment at a macroscopic level. It comprises interconnected sensor nodes that sense a physical parameter of interest. The sensed data is first digitized and then transmitted over a multi-hop network to a base station or sink. The main challenge of data collection in a wireless sensor network is to keep the transmissions volume within the limited bandwidth of the sensor nodes. Several efficient in-network processing techniques are developed to reduce the network traffic. These techniques are based on the fact that the raw data sensed by the sensor nodes are often compressible.

This thesis focuses on efficient sampling techniques based on the theory of *compressed sensing*. Compressed sensing or *compressive sampling* is a lossy signal compression technique for robust and efficient data collection by applying a simple network coding mechanism. The main advantage of compressed sensing over the existing methods is that it guarantees balanced load on the sensor nodes in a normal operation of the wireless sensor network. This effectively avoids exhaustion of overloaded sensor nodes.

In this thesis, we extend compressed sensing techniques for wireless sensor networks in the following ways:

Reordering for better compressibility: The effectiveness of compressed sensing depends very much on the compressibility of the sensed data. The more compressible the raw data is, the less transmissions are needed to collect the data. We show that compressibility is affected by the order or permutation of the samples. The samples recorded by a wireless sensor network are conventionally indexed by the sensor node id's. This indexing does not necessarily lead to the most compressible order of the samples. We propose an algorithm that maps the *physical* indexing of the samples to a *logical* indexing that is more compressible. This mapping does not require reprogramming or relocating the sensor nodes.

Spatiotemporal compressive sampling: Several studies show that the data sensed by a wireless sensor networks are both *spatially* and *temporally* compressible. We propose an extension to compressive sampling that takes advantage of *spatiotemporal* compressibility of the sensed data.

Handling link and node failures: In practice, the sensor nodes often experience occasional failures or link disconnections. We propose a novel variation of compressed sensing that is more tolerant to network perturbations. Our method detects the sensor nodes that are facing node or link failures and isolate their sensor reading to preserve the accuracy of genuine data.

Data dissemination via network coding: We introduce a novel network coding method that disseminates a particular linear combination of the sensed data to all sensor nodes of a wireless sensor networks. We show that the originally sensed data are recoverable from *any* arbitrarily chosen sensor node that receives an error-free linear combination. Our dissemination allows accessing the globally sensed data from any sensor node by performing local data exchanges.

Kurzfassung

Ein Sensornetz ist ein makroskopisches Sensoriksystem zur Überwachung und Messung von bestimmten physikalischen Parametern (Temperatur, seismische Wellen, etc.) einer Umgebung. Es besteht aus mehreren miteinander verbundenen Sensorknoten, die zusammen ein Multi-Hop Netzwerk bilden. Da die Netzwerk-Bandbreite und die Rechenleistung der Sensorknoten sehr eingeschränkt sind, werden spezielle Datensammlungstechniken benötigt, um eine große Datenmenge effizient bearbeiten zu können. Diese Dissertation konzentriert sich auf räumliches und zeitliches Sampling in drahtlosen Sensornetzen mittels einer verteilten Implementierung der *Compressive Sampling* Theorie. Compressive Sampling (auch *Compressed Sensing* genannt) ermöglicht eine robuste und effiziente Datenerfassung, ohne auf eine hohe Rechenleistung der Sensorknoten angewiesen zu sein. Die Besonderheit von Compressive Sampling ist die gleichmäßige Lastverteilung auf alle Sensorknoten, wodurch die vorzeitige Erschöpfung einzelner Sensorknoten verhindert wird. In der vorliegenden Dissertation erweitern wir die state-of-the-art Techniken von Compressive Sampling wie folgt:

Neu-Indizierung der Sensorknoten für bessere Komprimierbarkeit: Ein Algorithmus wird vorgestellt, der den Sensorknoten Indexe mittels eines neuen Schemas zuweist. Unter Anwendung der neuen Zuordnung sind die Daten besser komprimierbar wodurch sich die Effizienz von Compressive Sampling verbessert.

Raumzeitliche (spatiotemporal) Erweiterung von Compressive Sampling: Die Sensordaten wiesen meist Komprimierbarkeit sowohl in der Raumdomäne als auch in der Zeitdomäne auf. Wir stellen das Konzept von *Sliding Sampling-Window* vor, wodurch Compressive Sampling eine bessere Performance erreichen kann. Unser Konzept basiert auf einem virtuellen Sampling-Fenster, das mit der Zeit verschoben wird, um die zeitlichen und räumlichen Sensordaten zu sammeln. Unsere neue Methode hat die besondere Eigenschaft, dass abnormale oder fehlerhafte Sensorwerte sofort erkannt werden.

Behandlung von defekten Sensorknoten und fehlerhaften Verbindungen: Netzwerk-Störungen beeinträchtigen die Komprimierbarkeit der Sensordaten und die Performance von Compressive Sampling. Wir entwickeln neue Methoden zur Fehlerbehandlung, die sowohl gelegentliche als auch dauerhafte Fehler erkennen können. Die Sensorknoten werden nicht aktiv an der Fehlerbehandlung beteiligt. Stattdessen werden die fehlerhafte Daten durch eine besondere Nachbearbeitung erkannt und herausgefiltert. Auf diese Weise bleibt die Implementierung auf den Sensorknoten unkompliziert.

Datenverbreitung in Sensornetzen durch Netzwerkcodierung: Datenverbreitung bezeichnet sich auf ein Verfahren, das die gesamten Sensordaten für jeden Sensorknoten verfügbar macht. Wir präsentieren eine neue Netzwerkcodierung, die alle Sensordaten (unter bestimmten Nebenbedingungen und Qualitätsanforderungen) für jeden Sensorknoten verfügbar macht, nachdem eine Reihe von lokalen Datenaustauschoperationen zwischen benachbarten Sensorknoten durchgeführt wurde.

Acknowledgements

I extend my deepest gratitude to my advisor, *Prof. Dr. Neeraj Suri* for his support in pursuing the Ph.D. program and for everything I learned from him to follow the right path in the labyrinth of research. Without his guidance and encouragement, writing this thesis would never be possible. I also want to thank *Dr. Abdelmajid Khelil* who motivated starting this work and helped me in taking the initial steps and also guided me through the further stages.

I would like to thank my friends and colleagues at the DEEDS group with whom I spent a pleasant time and enjoyed working with them. Many thanks to *Azad, Piotr, Oliver, Faisal, Vinay, Brahim, Daniel, Stefan, Thorsten, Matthias, Ute, Sabine, Peter, Hamza, Robert, Kubilay, Tsvetoslava, Jesus, Ahmed, Heng, Dan, Marco*, and *Hatem*. They created an excellent atmosphere in the group and helped me a lot from the very first day of arriving in Darmstat. Also, I really appreciate their kind assistance for settling down in Darmstadt and the support which lasted all through the years that I was working on this thesis.

I want to thank my parents and my family for their love and for all the good things that they taught me. Also, many thanks to *Mitra* and her family for the great help in the time of moving from Iran to Germany. I also want to thank all of my teachers during all levels of my education.

Finally, a very special thank goes to my beloved wife *Mina*. I am so thankful for your love and understanding. Thank you for the encouragement and for supporting me in all of those hard days. Thank you for giving me hope, courage and the reason not to give up.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction and Problem Context

Wireless Sensor Networks (WSNs) are distributed sensory systems for large-scale monitoring of physical parameters of interest such as seismic vibrations, temperature, humidity, light intensity, radiation level, etc. Sohraby et al. [2007]. A WSN consists of battery-powered Sensor Nodes (SNs) that communicate with each other over a wireless medium. Apart from battery-powered SNs, there are energy-harvesting SNs that acquire their required energy from the environment, e.g., through photo-voltaic cells. In either case, the power consumption of the SN is very strictly limited. Due to energy constraints, the communication range is limited; hence, each SN can only exchange information with a few neighboring nodes Mahfoudh and Minet [2008]. Therefore, the sensed data is often transferred over multiple hops of interconnected SNs to reach a destination node. WSN is often referred to as a *bridge* between information systems and the physical world Elson and Estrin [2004]. WSNs have varied applications such as surveillance, monitoring, industrial automation, home control, etc.

The primary objective of a WSN is to deliver the sensed data to a base station or *sink*. Sink is a dedicated node with sufficient computational resources and power that post-processes the data and prepares it for the *end user*. Throughout this thesis, the *end user* or simply *user* refers to any external entity that is the final consumer of the WSN data. This entity can be a person or another software that processses the incoming data.

Each SN is typically equipped with a sensing device that records the physical parameter of interest. These fine-grained sensor acquired values are quantized and converted to digital data and transmitted over the wireless network to the sink. The SNs often possess limited computation, power and bandwidth Polastre et al. [2005].

It is crucial to efficiently transmit the sensed data over the resource limited SNs. Otherwise, the battery of the SNs will deplete very rapidly. This causes exhausted nodes that may lead to disconnected or partitioned networks. Network partitioning is extremely costly, because the exhausted nodes must be redeployed or replaced in order to make the WSN operative again. Redeployment of a WSN or some SNs is often not desired, especially when the WSN is deployed in a hard-to-reach location, such as a desert, forest, mountains, etc. WSNs are ideally expected to be installed once and operate for several months or years. Energy harvesting SNs do not face the problem of battery exhaustion. However, the power provided by the energy source (e.g. photo-voltaic cell) is usually very limited. This constraint limits the bandwidth and processing power of the SN.

1.1 Problem statement

This thesis focuses on distributed data compression techniques in WSNs using network coding techniques that are based on compressed sensing. Compressed sensing is an efficient method to acquire samples from a sensory system when accessing all individual samples is costly. Compressed sensing applies very well to distributed sensing problems like WSNs where each sample is recorded at a distant location and requires several message exchanges to be collected at a central base station. Within this context, we target the following problems as described in the coming subsections.

1.1.1 Reordering for better compressibility

Compressed sensing is based on the linear transformation of the sensed data under the condition that the sensed data must be compressible under a linear projection. The data is mathematically represented as a vector of n real numbers for a WSN consisting of n SNs. Each item of this vector corresponds to a value sensed by a SN. Modeling the sampled data as a real vector is a common approach for representing the sensed data. Sampling takes place either in spatial or temporal domain. In some applications, the spatiotemporal sampling acquires the data in both space and time dimensions.

Note that throughout this thesis we may use both forms *spatio-temporal* and or *spatiotemporal* sampling and both refer to sampling or data acquisition taking place in both space and time domains.

Temporally sensed data are bound to an order that is determined by the time that samples are recorded. The sample recorded at time t always precedes the sample recorded at time $t + 1$. The order of the temporal

samples is defined by an independent physical parameter, i.e., time. The order or permutation of the spatial samples usually depend on the placement of the individual sensors. For example, a camera records the light intensity of different wavelengths using a matrix of sensitive photo-cells. The order of the pixels is conventionally fixed and permuted form left to right and from top to bottom, representing a bitmap picture. If a WSN possesses a regular topology, for example when all of the SNs are placed on a straight row, then the order of the samples is normally defined by the order of the SNs on this row.

If the SNs of the WSN are randomly distributed in the network, no regular structure or topology is provided at the initialization of the WSN. The WSN is often deployed to be self-configuring. The SNs build an ad-hoc network to collect the data from across the network. In particular, when the SNs are scattered in the environment, the order of samples has much less correlation to the geographical locations of the SNs.

The conventional ordering of the samples recorded by a WSN is given by the id's of the SNs. So, the SN 1 is *conventionally* reading the sample 1 of a data vector, SN 2, the sample 2, and so on. Nevertheless, the SN 1 and SN 2 could be placed far from each other and their sensor readings might have not any correlation to each other. Note that correlation plays an important role in the success of the compression techniques such as compressed sensing ElBatt [2009].

We define a systematic *labeling* of SNs in a WSN that overlays the conventional labeling of the SNs by their id's. Through a mapping between our labeling and the SN id's, we achieve a new ordering (permutation) of the samples that is more compressible under a transform compression. This finally leads to less measurements that are required for data collection techniques based on compressed sensing. More interestingly, when a suitable mapping is found for a certain time instance of the recorded data, it still produces a more compressible reordering of the samples for the next time instances, compared to the typical ordering of the samples by SN id.

1.1.2 Spatiotemporal compressive sampling

The physical phenomena sensed by a WSN often shows compressibility in both space and time dimensions Vuran et al. [2004]. The data sensed by two adjacent sensor nodes are expected to be correlated, and hence, compressible Duarte et al. [2005]. Furthermore, the recorded values by a single sensor node over consequent time instances are also often compressible. A compression method that benefits only from spatial compressibility is called *spatial compression*, i.e., compression only in space domain. A compression

method that only considers compressibility in time domain is called *temporal compression*. Most of compression techniques achieve much higher efficiency when they are applied on both spatial and temporal data. As an analogy, consider a typical digital video which applies spatiotemporal compression of raw video file, and thus, achieves a much higher compression ratio than when each frame is compressed as a separate picture.

Compressed sensing in its original form for WSNs allows an easy implementation of spatial compression. It can be also easily extended to spatiotemporal compression. One drawback of extending the compression to temporal domain is that, the raw data need to be sampled over a time frame in order to apply an efficient encoding and compression. The width of this time frame greatly influences the delay of transmitting the data. A longer time frame improves the performance of temporal compression. Particularly in compressed sensing, it is desired to apply the coding over a larger sequence of data as the communication cost grows logarithmically with the size of data. On the other hand, longer time frames require that several samples over consequent time instances to be recorded first and then the compression takes place on those data. Delay-tolerant applications of WSNs such as long-term environmental monitoring can benefit the most from temporal compressibility. However, when an abnormal sensor reading is encountered, this can be detected after a certain delay. This delay is proportional to the length of the compression time frame, because the temporally compressed data must be first transmitted and unpacked at the sink.

We provide a solution that benefits from the efficiency of spatiotemporal compressed sensing and avoids delays in event detection. By introducing the concept of sampling window, we apply the temporal compression iteratively on time frames that are sliding over the sequential data over sequential time instances. While taking advantage of temporal compressibility with wider sampling windows, our solution can also detect the abnormal sensor readings in time. A larger sampling window increases the efficiency of our proposed method. At the same time, the penalty of selecting a larger sampling window is not a higher delay. The abnormal events are detectable while normal data collection is taking place. Only in case of failures, it takes a longer time for a larger sampling window to be reinitialized.

1.1.3 Compressed sensing in presence of link and node failures

One advantage of compressed sensing over traditional transform compression methods is its robustness to abnormal sensor reading. While most of com-

pression techniques rely on a presumed structure or statistical distribution of the sensed data, compressed sensing only requires the compressibility of the data under a linear transformation. Enhanced variations of compressed sensing are capable of detecting abnormal sensor reading without affecting the efficiency of data collection Luo et al. [2009].

We propose a solution for applying compressed sensing in a WSN in presence of link and node failures. Our solution is based on implicit distinguishing between erroneous and genuine data at the sensor level and detecting it after post-processing at the sink. We know that each sensor node is capable of sensing a particular range of data. For example, a temperature sensor is designed for sensing temperatures between a lower and upper bound. We add a real number as an offset to these lower and upper bounds. Therefore, the valid range of sensor readings is elevated by the offset. Our solution is then able to detect the node or link failures by finding the samples that are suddenly dropped to zero because they were not able to transmit their data due to an internal or external failure. We show that, this method can effectively detect the failing sensor nodes. Furthermore, we propose an enhancement to the compressive data gathering in WSNs (see Luo et al. [2009]) that is able to isolate the failing nodes from data collection.

1.1.4 Network coding for data dissemination in WSNs

Data dissemination in a WSN refers to distribution of the sensed data to all sensor nodes of the WSN. Dissemination is especially useful for applications where each node of the network requires the full knowledge about the operational environment. For example, when a distributed control is to be performed to control the global quantity of a physical phenomena. Such scenarios can be most efficiently handled when the global knowledge about the operational environment is accessible by all nodes of the network. This scenario is often studied in Wireless Sensor and Actuator Networks (WSANs). A WSAN is comprised of both sensing and actuating nodes. In some applications, the nodes have a hybrid role of sensing and actuating. Sensing nodes acquire the information about the environment. Actuating nodes try to change or maintain the attributes of the environment in an efficient way to achieve a desired state of the environment.

Dissemination allows each node of the network to take the role of the sink node. We propose a method for data dissemination based on compressed sensing that takes the compressibility of the data into account. As discussed earlier, the data sensed by a WSN is expected to be compressible. Our proposed method takes advantage of this compressibility to perform the dissemination more efficiently. Our solution has also a very low computation

complexity for the sensing nodes. By exchanging local information between neighboring nodes, a set of linear combinations of the sensed data is distributed across the network. We show that, with our special combination of source- and network-coding, it is possible to recover the originally sensed data from the linear combinations that are received by any arbitrarily chosen node of the network. Depending on the processing power of the nodes, some of them are chosen for executing the recovery algorithm. Another possibility is that a mobile node that has enough processing power fetches the linear combinations from one or more sensor node(s) in its vicinity. The mobile node can access the global state of the environment after applying the recovery step. The latter scenario is especially useful in mobile sink scenarios such as a combination of a robot and WSNs.

1.2 Thesis contributions

Within the context of the problem statements, we list the following contributions of this thesis.

- **Contribution 1 – Reordering technique for better signal compressibility and a polynomial-time algorithm to find a more compressible permutation of the samples**: Given a set of sensed data and a compressive basis Ψ, the objective of our algorithm is to find a permutation of the samples, such that the sensed data are more compressible in Ψ. The algorithm is executed at a time instance t and the WSN continues sensing the data at time $t + 1$. The SNs are relabeled according to the more compressible permutation that was found in the previous time instance. Enhanced compressibility of the reordered permutation proves to be extended over longer time spans. Consequently, the more compressible ordering leads to a better performance of compressed sensing at time instances after t.

- **Contribution 2 – Concept of sliding sampling window for spatio-temporal compressed sensing in WSNs**: A sliding window for sampling over time is conceptualized. Our concept aims to exploit temporal as well as spatial compressibility of the sensed data. A larger sampling window makes the compressed sensing mechanism to operate more efficiently because the communication cost of compressive sampling grows logarithmically with the size of the spatio-temporal signal. On the other hand, a too large sampling window requires a longer time to get filled (initialization) and also longer time to refill (reinitialization) when the network encounters an unrecoverable failure. The time

for filling (buffering) is required only once during the initialization of the network. Afterwards, the data is delivered to the user in time. In our proposed model, the delay of delivering the sensed data does not grow drastically. Furthermore, our evaluations show that our method is able to detect abnormal sensor readings and report them in time.

- **Contribution 3 – Methods to detect the failing nodes and isolate the negative effect of their abnormal sensor reading on the compressibility of data**: It is known that abnormal sensor readings usually degrade the performance of compression algorithms. We consider SN failures as an abnormal event and design a sensing mechanism that models a failing node as an abnormal value in the sensed data. If the compression algorithm relies on the statistical distribution of data to be preserved during runtime, it suffers most from sensor node failures and or abnormal sensor readings. CS shows more robustness to abnormal sensor readings, because it does not assume a statistical distribution of the sensed data before sampling. However, CS is also affected by the negative consequences of abnormal sensor readings, because abnormal values in the signal often degrade the compressibility of the signal. We propose an *event detection* mechanism for our spatiotemporal sampling window mechanism. Our method is based on recovering the data using an over-complete dictionary. The first step is to detect the abnormal values with the help of data recovery in an over-complete dictionary. In the first step, we inform the user about the failing nodes. In the next step, the failing nodes are excluded (isolated) from the recovery process, and the sensed data are reconstructed again. We observe a much higher quality of the recovered data after excluding the missing nodes.

- **Contribution 4 – Novel network coding mechanism based on compressed sensing for efficient signal dissemination in WSNs**: The network coding technique introduced in this thesis allows dissemination of the sensed data to all sensor nodes without collecting it at a central point. Our coding mechanism only involves the exchange of numerical values and performing simple arithmetic operations, i.e., addition and multiplication. We show that, by performing the proposed network coding technique, each SN receives a set of linear combinations of the sensed data. Our numerical experiments show that these linear combinations correspond to a measurement transform that allows recovery of the original data at any node. Our dissemination framework called *Comprensus* allows accessing a globally sensed signal by fetch-

ing a small amount of information from a node in the vicinity of the receiver.

1.2.1 Publications

The topics discussed in this thesis and the research outcomes are published in the following papers. Some contents of these publications are partly included in the thesis:

- Mohammadreza Mahmudimanesh and Neeraj Suri, "Agile sink selection in wireless sensor networks", 11th IEEE International Conference on Sensing, Communication, and Networking (SECON 2014), pp. 390–398, IEEE, 2014.
- Mohammadreza Mahmudimanesh and Neeraj Suri, "Robust Compressive Data Gathering in Wireless Sensor Networks with Linear Topology", IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2014), pp. 179–186, IEEE, 2014.
- Mohammadreza Mahmudimanesh, Amir Naseri and Neeraj Suri, "Efficient Agile Sink Selection in Wireless Sensor Networks Based on Compressed Sensing", IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2014), pp. 193–200, IEEE, 2014.
- Mohammadreza Mahmudimanesh, Abdelmajid Khelil and Neeraj Suri, "Compressive Sensing for Wireless Sensor Networks" in "Intelligent Sensor Networks: The Integration of Sensor Networks, Signal Processing and Machine Learning", pp. 379–395, CRC Press, 2012.
- Mohammadreza Mahmudimanesh, Abdelmajid Khelil and Neeraj Suri, "Balanced spatio-temporal compressive sensing for multi-hop wireless sensor networks", IEEE 9th International Conference on Mobile Adhoc and Sensor Systems (MASS 2012), pp 389–397, IEEE, 2012.
- Mohammadreza Mahmudimanesh, Abdelmajid Khelil and Neeraj Suri, "Reordering for better compressibility: Efficient spatial sampling in wireless sensor networks", IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2010), pp. 50–57, IEEE, 2010.

Topics of this thesis also partly contribute to the following papers or are related to them, though the publications are not a direct research outcome of this work:

- Azad Ali, Abdelmajid Khelil, Neeraj Suri and Mohammadreza Mahmudimanesh, "Adaptive Hybrid Compression for Wireless Sensor Networks", ACM Transactions on Sensor Networks (TOSN), vol. 11, no. 4, p. 53, ACM, 2015.

- Mohammadreza Mahmudimanesh, Abdelmajid Khelil and Nasser Yazdani, "Map-based compressive sensing model for wireless sensor network architecture, a starting point", Mobile Wireless Middleware, Operating Systems, and Applications Workshops, pp. 75–84, Springer Berlin Heidelberg, 2009.

1.3 Thesis structure

Chapter 2 reviews the state of the art for efficient acquisition of spatial, temporal and spatiotemporal signals. We focus on data reduction techniques, and then, we narrow down our study to distributed compression techniques for WSNs.

Both chapters 2 and 3 discuss mainly the state of the art techniques related to this thesis. Chapter 3 is focused on CS-based techniques. It also gives a detailed description of the system model that is considered in this thesis.

Chapter 3 focuses on the CS theory and elaborates on its foundations. The concepts of the CS theory that are needed for the rest of the thesis are described in more details. We also discuss some of the existing implementations of the CS methods for WSNs. Parts of our solutions are based on the frameworks that are reviewed in this section or are enhancements to them.

Chapters 4 to 7 give technical details of our contributions 1 to 4 as listed in Section 1.2. Chapter 4 contains the technical details of Contribution 1. Chapter 5 details Contribution 2. Chapter 6 describes Contribution 3. Finally, Chapter 7 focuses on our Contribution 4, namely the Comprensus data dissemination technique.

Chapter 2

State of the Art and Practice

In this section, the state of the art techniques for gathering and disseminating the WSN data is reviewed. Since the study field of WSNs is broad, we first need to focus on the scope of this thesis and discover the most relevant related works. This thesis mainly focuses on the efficient data processing techniques to make the data available to a specific user or third-party application. Unless we specifically mention it, we assume that the routing, network management and similar practices that are not directly related to the sensed data, are performed by any suitable existing method that is designed for such tasks.

In particular, we assume that the SNs are able to discover their neighboring nodes. Also, during the network establishment, the SNs know to which neighbor they have to send in order to deliver their data over multiple hops to reach the base station. For example, in the context of data collection, it is usually needed to make a collection tree before starting the data collection phase. For example, the spanning tree creation method by Tan et al. Tan and Körpeoğlu [2003] can be applied to form a spanning tree over the network.

Suppose that a spanning tree is formed for the WSN and the sink is positioned at at the root of this tree. A trivial baseline solution to collect the data at the sink is to transmit the sensed data hop-by-hop towards the sink. However, this obviously overloads the nodes closer to the sink. Each node has to transmit its own data and forward the data from the nodes in lower levels of the spanning tree. Therefore, when a node is closer to the sink, it has much more data to forward than to transmit. The overloaded nodes will eventually run out of battery and cannot transmit and or forward the data. Consequently, the spanning tree must be rebuilt to recover the connectivity of the network. When too many nodes are overloaded, it is not possible anymore to recover the network connectivity. In this case, the network is partitioned at the overloaded nodes. When network partitioning

happens, a large amount of sensed data will be inaccessible, since they cannot be delivered to the sink.

It is crucial to reduce the amount of the data that is being transmitted by the SNs. Radio transmission is a major energy drain of a SN. By reducing the amount of transmissions, SNs consume less energy and can prolong their lifetime. In this section, we categorize the data reduction and compression techniques based on different criteria. Then we narrow down our study to the categories that are most relevant to the approach that is presented in this thesis. This provide a good background and an insight into the context of the problem that is targeted in this thesis.

2.1 Data reduction approaches

Data reduction techniques mainly rely on two factors:

- *Compressibility of the sensed data*: If some pattern or structure is detected in a dataset, a shorter summary of the data is extracted by the compression algorithm. This shorter piece of information can be more efficiently stored or transmitted than the raw data.

- *User's accuracy and timeliness requirements*: Depending on user's accuracy and granularity requirements, data reduction techniques can be optimized for better efficiency. For example, if the user is only interested in the arithmetic average of the sensed data, a large amount of raw data can be reduced to a single data item with minimal computation complexity. Timeliness requirement can play an important role as well. When longer data collection periods are allowed by the user, more temporal correlation of the data can be exploited by the compression algorithm to further reduce the size of the data.

Aggregative operations such as averaging are another type of data reduction methods. However, we want to distinguish such data reduction methods from compression. For this reason, we use two different terms to refer to these techniques and classify them by two criteria.

- *Aggregation* functions are non-reversible functions. It is not possible to retrieve the original data from the outcome of an aggregation function. For example, if a is the average of n sensor readings, it is not possible to recover the individual sample values from a as soon as $n > 1$.

- *Compression* processes are generally reversible processes. Compressed data can be partly or completely decompressed to access the originally recorded data.

Throughout this thesis, either of the terms compression or data reduction refer to reversible processes that reduce the amount of data. Aggregation methods as a subset of data reduction technique are not the topic of this thesis.

Let D_{raw} be a finite ordered list of real numbers that contains all sensor values recorded at a certain time instance by the WSN. A compression function γ takes D_{raw} as input and reduces it to a smaller data set $D_{compressed} = \gamma(D_{raw})$ such that size of $D_{compressed}$ is less than D_{raw}:

$$D_{compressed} := \gamma(D_{raw}) \text{ such that } |D_{compressed}| < |D_{raw}| \tag{2.1}$$

Decompression process is then the inverse process γ^{-1} that takes $D_{compressed}$ as input and returns the decompressed data set $D_{decompressed}$.

$$D_{decompressed} := \gamma^{-1}(D_{compressed}) \tag{2.2}$$

2.1.1 Lossy and lossless compression

Depending on the accuracy of the decompressed data, compression schemes can be classified into two categories: Lossless and lossy.

- *Lossless compression*: No data is lost after applying the compression and decompression functions to the original data, i.e., $D_{decompressed} = D_{raw}$. Lossless compression is usually used in archiving and transmission of large sensitive data such as software application files or medical images. It is widely used in commercial compression softwares. Some of the most commonly used lossless compression algorithms are Run-Length Encoding (RLE), Lempel-Ziv-Welch (LZW) and Lempel-Zim-Markov chain algorithm (LZMA) Parekar and Thakare [2014]. These algorithms are used in many compression applications and several file formats like Graphics Interchange Format (GIF), Portable Network Graphics (PNG), ZIP, etc.

- *Lossy compression*: Data loss or inaccuracy of the decompressed data is allowed to a certain level according to the accuracy requirements of the user, i.e., $D_{decompressed} \approx D_{raw}$. Lossy compression allows unimportant data to be lost in sake of more important data. Most of lossy compression techniques involve transforming the raw data to some other domain like frequency domain or wavelets domain Graps [1995].

The distributed compression techniques introduced in this thesis are all lossy compression. There is a tradeoff between accuracy of the data that

is delivered to the end user and the communication cost for the WSN. Our techniques allow tuning this tradeoff to suite specific requirements of the end user regarding energy preservation, accuracy and timeliness of the received data.

2.1.2 Source coding and network coding

Compression techniques for WSNs demand a different implementation from other applications of data compression, mainly because the compression takes place in a distributed manner. The compression function γ introduced in Equation 2.1 represents an ideal theoretical representation of a compression scheme that is applied at once on the whole data set. Even when the whole raw data set is available to the compression algorithm, the compression usually takes place on smaller chunks of data that fit into the memory of the computing machine. Apparently, more correlations within the raw data are observable if the whole data is loaded into the memory and the compression algorithm can access the whole data set at once. However, practically this ideal approach can not be implemented for larger amounts of data in order of hundreds of megabytes or gigabytes. It requires a lot of memory and processing power to compress large amounts of data as single integral dataset. In practice, even if the whole dataset is available beforehand, most of compression algorithms work with much smaller chunks of data and process each of them separately.

In a WSN, it is even more challenging to compress the sensed data. The raw data (sensed values by the SNs) are stored in SNs across the whole network. Acquiring each and every sample requires fetching the sensed value from the SN. This is a very resource-intensive operation, since the sensed data often need to be forwarded over several hops to become accessible by a central compression algorithm. Therefore, the compression schemes for WSNs differ for instance from audio or video compression in which the raw picture or sound is accessible directly by the processing unit that runs the compression algorithm. In audio and video compression, the source of raw data (image sensor or microphone) is located close to the processor that performs the compression. In most cases they are mounted on the same device, such as a digital camera or a digital voice recorder. In contrast to these applications, the data sources of a WSN are distributed in a large environment. Therefore, the data compression techniques have to be implemented in sensing and or network communication layer. Distributed data compression in WSNs is thus closely related to both source coding and network coding.

- *Source coding* refers to transformation of data at the SNs. As an ex-

ample, consider a WSN that is programmed to report the position of the most rapidly changing sensor values. The SNs thus apply a simple source coding operation by differentiating the consequent sensor readings and reporting the difference instead of the the actual sensor data. Each node compares the received difference by its own calculated value and sends the larger value along with the corresponding SN id to the next hop.

- *Network coding* refers to transformations that are performed on the data while it is traversing the WSN. In contrast to source coding that only affects the values sensed by a single SN, network coding affects also values sensed by other SNs.

Source coding and network coding play an important role in effectiveness of compression techniques for WSNs. By reducing the data at the SNs, a large amount of raw data traffic is already removed from the network. Network coding helps further to reduce the amount of data while it is being transmitted hop by hop through the network. It brings no value to compress the data of a WSN after it arrives its destination. The final destination of the WSN data (usually the sink node) has often much higher computation and storage resources than the SNs. It is too late to apply data reduction on the data that is received by the sink. The most valuable resources of a WSN, i.e., the SNs power and bandwidth will be consumed very rapidly when no or little data reduction is performed by source- or network-coding. Data reduction techniques introduced in this thesis apply both source coding and network coding to achieve a more efficient data collection scheme in WSNs.

2.1.3 Transform compression

Compressibility of the data may not be apparent at the first glance. However, when the data is transformed using a transformation function, it can be seen that the the data are compressible with one of the known encoding techniques. The transformation must be reversible in order to be able to recover the original data. Often, the transformation is performed by a linear transform on discrete data. A linear transform can be represented by a matrix multiplication. The data is represented by a vector and a transformed vector is the result of multiplying a transform matrix by the data vector. For a suitably chosen transform matrix, the transformed vector is expected to be more compressible than the original data. The inverse of the transform matrix must exist in order to be able to apply the decompression.

Transform compression forms the basis of many signal compression techniques. Discrete Cosine Transform (DCT) and Fourier transform are some of

the most commonly used linear transformations for compressing the signals. Wavelet transform is another commonly used transformation for compression Graps [1995]. Transform compression is based on the fact that most signals recorded from natural phenomena are highly compressible under DCT or Fourier transform. Being compressible means that, most of the data items in the transformed data vector are nearly zero. Only a few of the items in the transformed vector have a significant value. Transform compression is based on finding the most significant (largest) values and storing those values along with their location (their index in the transformed data vector).

Lossy decompression of the transformed data is performed by restoring a few of the most significant values and keeping the rest of the compressive coefficients zero. By applying the inverse transform on this vector, we are able to recover the original data with a reasonable accuracy. The compression and decompression process is lossy, because the recovered data differs from the original data by a difference proportional to the magnitude of least significant items which were *thrown away* during the compression process. There is a tradeoff between the compression ratio (the ratio of the number of most significant values kept to the size of the whole data) and the accuracy of decompressed data. Depending on accuracy requirements, transform compression can be easily tuned to achieve a desired compression ratio while delivering the desired degree of accuracy.

Lossless compression keeps all of the transformed data and applies a standard entropy coding technique such as Hufmann coding, or run-length coding to further reduce the stored or transmitted data. The decompression comprises then of two phases. First, encoding string of data is decoded (decompressed). Second, the inverse transform is applied on the result of the first phase to recover the original data. In a lossless compression, the original data is fully recovered. Lossless compression achieves less compression ratio than lossy compression.

2.2 Data compression in distributed sensory systems

When we look at many sensory systems, we usually observe three different functions.

1. *Sensing*: A sensor records the physical value of interest, like light intensity, sound, pressure, temperature, etc. and an Analog-to-Digital Converter (ADC) quantizes the recorded value and outputs a binary number that reflect the sensed value.

2. *Transmission*: The sensor recordings are transmitted periodically over a communication channel. Depending on the technology used for transferring data, this process entails different levels of transmission cost. For example a camera connected directly to a computer, has a fast, accurate and cheap (low power, inexpensive) communication channel for transferring data. In such trivial cases, raw data is sent without much processing and the receiver (in this example, the computer) is responsible for further computation steps. On the other hand, in several sensor network applications, the raw data is to be transmitted over a noisy wireless channel. For example in a WSN, a vast amount of data has to be transmitted over a noisy wireless channel. Because of collision, congestion and inevitable errors, some of the packets are dropped, causing more and more retransmissions. Such a communication channel is too expensive for the sensor nodes (SNs) since each transmission consumes a lot of node's battery power. When the communication is too expensive or the amount of raw data is far more than channel capacity, the raw data must be compressed before or during the transmission to utilize the communication line more efficiently.

3. *Processing and Storage*: The raw data collected from the individual sensors is then processed by the sink or stored on persistent storage for future processing. The data collected over the network can be encoded, encrypted or compressed. Depending on the application of the sensory systems, a combination of compression, encryption or other special encodings are applied.

2.2.1 Compressibility of signals

Figure 2.1 shows an example of Fourier transform that is run on signal acquired from a temperature sensor that records ambient temperatures periodically LUC [2008]. Despite the signal itself in time domain does no appear to have a specific structure, its Fourier transform is quite compressible. We see that only very few Fourier coefficients have a large magnitude and most of other coefficients are quite negligible. One can devise a compression technique by keeping only the largest Fourier coefficients and discarding the other ones.

To show how efficient transform-coding compression techniques are, we do a simple numerical experiment on the time-domain signal of Figure 2.1. We represent the signal as a vector $\mathbf{f} \in \mathbb{R}^N$. Assume $\mathbf{x} \in \mathbb{C}^N$ is the Fourier transform of $\mathbf{f}$. We select k largest elements of $\mathbf{x}$ and set its other $N - k$ elements to zeros. We name the resulting vector $\mathbf{y}$. Now we do an inverse

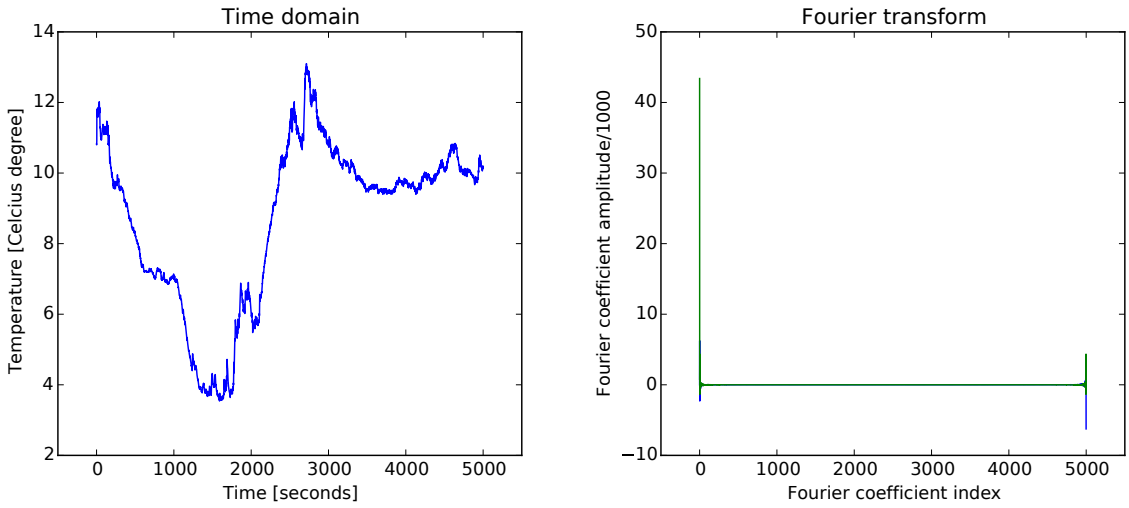


Figure 2.1: Compressibility of a natural signal under Fourier transform

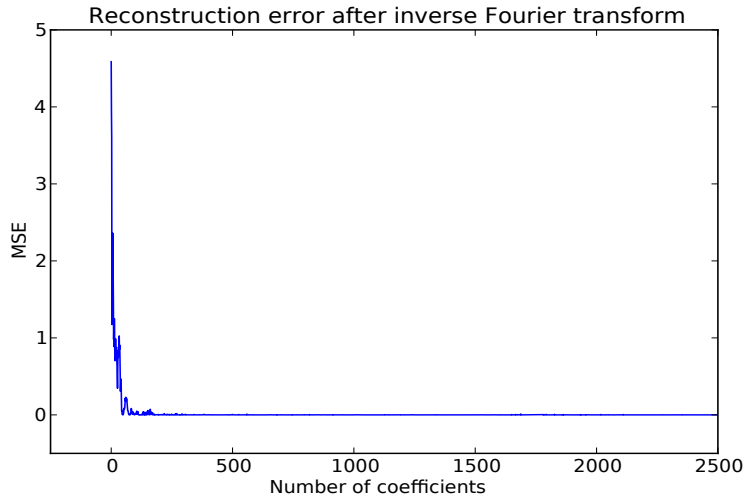


Figure 2.2: Reconstruction error by keeping the first Fourier coefficients

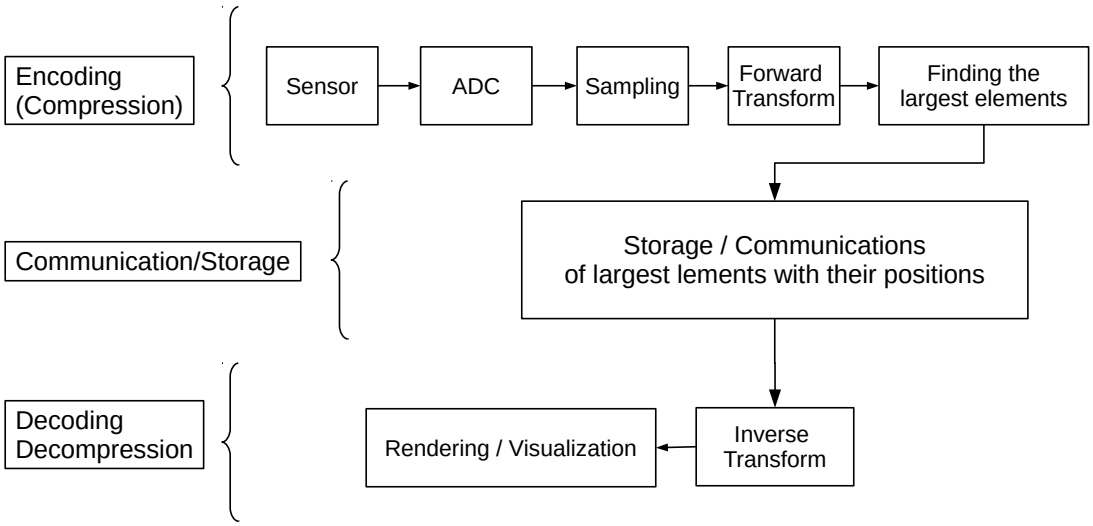


Figure 2.3: Traditional transform coding compression and decompression system

Fourier transform on $\mathbf{y}$ and get the vector $\mathbf{g}$, a recovered version of the original signal $\mathbf{f}$. Since the signal is to be recovered from incomplete set of information, reconstruction error is inevitable. We measure the error by calculating mean square error (MSE). Figure 2.2 illustrates that the recovery error rapidly decreases when we increase number of Fourier measurements. In other words, almost all energy of the time-domain signal of Figure 2.1 is concentrated on the few largest Fourier coefficients of its frequency-domain transform.

Many other signals acquired from natural phenomena, like sound, radiation level, humidity as well as more complex audiovisual signals recorded by cameras and microphones have a very dense support on a suitably chosen domain like Fourier, wavelet or DCT Broughton and Bryan [2011]. Therefore, we can depict the overall function of a transform coding system in Figure 2.3. Encoding part of the compression system illustrated in Figure 2.3 is a complex function that is run on a large amount of data. Note that after determining the largest elements of the transformed signal, negligible values are simply thrown away.

2.3 Distributed compression techniques in WSNs

Distributed implementation of the discussed compression techniques is not trivial. WSN is a distributed sensing system that requires data compression to keep the data transmission, and finally, the power consumption low. The level of compressibility of the sensed data is not known before accessing the whole set of data at once and at a single point. This is not possible in a multi-hop WSN, since each data item is sensed by a separate node of the network.

The classical system model of Figure 2.3 is not appropriate for distributed sensor systems like WSNs, mainly because the communication, sampling and encoding are taking place simultaneously in a WSN. Apart from that, the encoding/compressing complexity needs to be low, while the decoding/decompressing is often performed by the sink which has sufficient computing power. Depending on the computation complexity of encoder/compresser and decoder/decompresser we classify three different application classes of compression algorithms.

- *High compression ratio with complex algorithms and easy to decode/decompress data payload*: This variant has several applications in multimedia codecs (coder/decoders). The raw (uncompressed) data is often in form of audio and or visual files with high details and a very large size. In such applications, the objective is to store as much as possible details in a small information payload. Only high compression ratio allows, for instance, online watching of high definition videos on the internet. On the other hand, the decoding/decompression phase should not be resource intensive. Today's trend is to be able to play multimedia files on resource-limited mobile devices. As mentioned, this class of compression is suitable for multimedia applications.

- *Fair compression and decompression resource consumption*: Fairness for encoder and decoder is more of interest for verifying the security of a system in which compression of data in a communication protocol is allowed. When a protocol allows non-complex encoding and very complex decoding, an attacker can launch denial of service attacks by sending a compressed information package that causes a lot of computations on the receiver side to decompress the data.

- *Resource-constrained encoders and powerful decoder*: Distributed sensing and data collection is a good example for this category of compression methods. Unlike the other two categories, it is required to encode

and compress the data using the least amount of resources, while decoding can be done by a computation entity that possesses sufficient resources.

WSNs fit into the third category of the compression approaches mentioned above. The encoding takes place in the sensor nodes which are resource-limited. One important requirement for a compression algorithm in WSNs is to be very low complex at the encoder side. The complexity of decoding is not an issue in most configurations of WSNs, because the decoding takes place at a dedicated node outside the sensor network that performs the computations to recover or reconstruct the originally sensed data.

In this section, we discuss three general methods of data compression in WSNs, namely in-network compression, Distributed Source Coding (DSC) and Compressed Sensing (CS). Most of the compression techniques for WSNs can be considered as a specialization of these general methods. Some other works provide enhancements or extensions of these general approaches which are applicable to specific scenarios.

2.3.1 In-network compression

In-network compression refers to a wide range of compression methods that are implemented both on the sensor node level and network level. Assuming that the sensed data is compressible, the objective of in-network compression is to find an optimal combination of routing and encoding throughout the path that the data is traversing hop by hop to reach the sink. Most of the in-network compression techniques are based on transform compression. First, a suitable compressive transform needs to be chosen depending on the signal that is to be sensed by the WSN. DCT or Fourier transform is suitable for acquiring signals such as ambient temperature, humidity, etc. Wavelet transform better captures abnormal changes in the spatial data. Wavelet can be used to detect events in an environment or signals that have drastic variations or fluctuations.

In addition to the typical transformations such as DCT, Fourier or wavelet, Karhuenen-Loève Transform (KLT) is considered as a better transformation, since it theoretically achieves the optimal compressive basis for a given signal Gastpar et al. [2006]. While the transformation matrices of DCT and Fourier transform can be generated using a finite number of computations, there is no practical solution for generating the entries of a KLT matrix for a given signal profile Marco F. Duarte and Baraniuk [2012]. Regardless of the compressive basis that is used for in-network compression, the transmission of the data has to take place in a compressed form. The

compressed data are gathered over a spanning tree. The sink receives the compressed data and applies the decompression algorithm to extract the originally sensed data. Shen et al. Shen and Ortega [2010] introduce a data gathering technique based on transform compression.

To achieve the best compression ratio, transform compression must be applied to the complete set of data rather than on small portions of the sensed data. Therefore, when each node receives the compressed data, it has to perform a local decompression. It first extracts the raw data and appends its own data to the received data. Then the compression algorithm is applied again to build a compressed packet that is sent to the next hop.

One drawback of this approach is that the sensor nodes need to possess enough computation power and memory to decompress the partially received data, and then, append their own data. Furthermore, each SN requires a relatively powerful CPU and a relatively large amount of RAM to compress the data and send it to the next hop. Note that, a complex compression algorithm can be very heavy for the basic hardware of a typical SN. Some other SN hardware architectures provide more computation power, though with the cost of more energy consumption. The memory requirement can grow very high especially for the nodes closer to the sink. These nodes have a lot of descendants in the collection tree. Therefore, the compressed data packets that are received by these nodes will occupy a larger amount of RAM when uncompressed. The efficiency of compression algorithms depends very much on the amount of data that can be accessed centrally by the computation algorithm. This is mainly decided by the amount of data that can be loaded directly into RAM, because the compression algorithm has to process a buffer of data at once and then store the compressed buffer, in order to achieve the highest overall performance.

Assuming that the SNs possess enough computation power and memory, in-network transform compression still suffers from another drawback. The amount of computations performed by the SNs is not balanced across the sensor network. The nodes closer to the sink perform more computation and may require more RAM than the nodes that are in the middle of the collection tree hierarchy. The leaf nodes have the least computation and memory requirements, since they have only one data item and do not require to compress it at all. This leads to an inhomogeneous composition of the SNs in a WSN.

Inhomogeneity of the SNs makes the WSN very inflexible to configuration changes. It is desired that all SNs have the same hardware platform, otherwise the WSN becomes very inflexible to configuration changes. According to the homogeneity requirement, the SNs must be equipped with a hardware that can handle data compression and decompression in its worst case, i.e.,

very close to the sink node. Having a homogeneous network enables the user to reconfigure the WSN at any time. For example, the user may want to pick another location for the sink. This is only in a network with homogeneous SN hardware platforms possible without massive reconfiguration or relocation of the SNs. In the exemplified scenario, we see that a huge amount of computation power of the SNs remains unused. Only the nodes closer to the sink may fully advantage from their relative high-performance hardware.

We summarize the pros and cons of in-network transform compression as follows:

- **Pros**
  - *Higher compression ratio*: By applying the transformation directly on the sensed data, in-network compression often achieves a higher compression ratio.
  - *Suitable for low bandwidth applications*: In-network compression is one of the best solutions for scenarios where the bandwidth is very limited, but the computation power of the sensor nodes allow uncompressing, padding and re-compressing the data.
  - *Extensible to temporal domain*: Temporal and spatio-temporal compression are direct extensions of the spatial in-network compression. The same methods for spatial compression are applicable to temporal and spatio-temporal data collection without major modifications to the state-of-the-art algorithms.
- **Cons**
  - *Complexity*: Efficient implementation of the in-network compression algorithms such that they can be executed by the basic hardware of the SNs is challenging.
  - *Unbalanced computation load*: The computation complexity depends very much on the amount of data that is passing through the node. In particular, the nodes closer to the sink have to perform more computations and require more memory to process and forward the data.
  - *Being sensitive to packet failures*: Data communication takes place only in the compressed form. Therefore, any loss of connection between two nodes destroys a relatively large amount of data. Compressed data packets are more sensitive to packet loss, because more information is likely to be lost when a single packet is lost or a small part of the data is corrupted.

Similar to network perturbation and packet loss, abnormal sensor readings also have a degrading effect on in-network compression methods. Especially, when a non-robust compressive domain is selected for transformation. In situations where abnormal sensor readings and unexpected events are possible, transform bases such as wavelet transform are more suitable choices to be used by in-network compression methods.

2.3.2 Distributed source coding

Distributed Source Coding (DSC) is based on correlation between data sources in the WSN. If two data sources are correlated, it is more efficient to use a joint encoding and save the number of bits required for storage or communication of the data. Suppose that, sensor A requires 6 bits to store its sensed data and the values sensed by sensor B are having at most 1-bit distance from the values sensed by sensor A. The possible differences to the value sensed by sensor A belong to the set $\{[000000], [000001], [000010], [000100], [001000], [010000], [100000]\}$. Only three bits are required to store the *difference* between values A and B, since the index to each of these 7 states requires 3 bits for its representation. In this example, a joint encoder can achieve a compression ratio of (6+6):(6+3) by sending the actual value of sensor A and then sending the index to the difference pattern of sensor B.

Joint encoding requires the two sources to communication with each other and choose an efficient coding set. This is avoided using Slepian-Wolf coding Slepian and Wolf [1973]. Slepian and Wolf showed that even if the data sources separately encode the data without any inter-communication, it is possible to transmit the data with an encoding rate equal to the joint compression data rate. The initial form of Slepian-Wolf coding is proposed for lossless coding. Wyner and Ziv extended source coding to lossy compression Wyner and Ziv [1976]. Lossy compression is of particular interest for WSNs and multimedia sensor networks.

The main drawback of DSC is that it depends very much on the correlation model between data sources. The statistical model of correlations between different sensor reading must be known for DSC to work properly. Furthermore, the correlation model must remain unchanged during the operation of the WSN. These two requirements are hardly achievable in a WSN. The WSN has to sense values from all SNs and observe the physical parameter over a relatively long period to achieve an accurate correlation model of the sensed data. Given the resource constraints of the sensor nodes, it is usually impossible to achieve such an accurate correlation model. Without an accurate correlation model, DSC cannot perform efficiently and is prone

to data collection errors. Even if the correlation model is known for a given WSN, there is no guarantee that this model remains unchanged for the whole duration of WSN's runtime. When the environment changes, the correlation model between the sensed data changes as well and a new correlation model must be determined for DSC to continue correct data collection.

To summarize the above discussion, we list the pros and cons of DSC as below:

- **Pros**
  - *Low complexity of coding*: The SNs perform simple computations to encode the data. This is of great importance for resource-limited SNs that are equipped with basic and low-power hardware.
  - *Memoryless coding*: DSC allows data collection schemes with memoryless SNs. The SNs do not need to store any data and simply encode and transmit their data. This is a great advantage for multimedia sensor networks where a lot of raw data is produced every second that needs to be encoded and transmitted using low resources and in a short amount of time.
  - *Tunable robustness*: Both Slepian-Wolf and Wyner-Zif coding methods provide mechanism to increase the robustness of data collection scheme to errors and channel perturbations. Both methods provide a well-defined tradeoff between sampling rate, compression ratio and coding strength. This allows the user to fine-tune the DSC according to application requirements and the conditions of the operational environment.
- **Cons**
  - *Need for accurate correlation models*: Accurate correlation models such as a covariance matrix of all sensor readings can be only calculated when a large amount of data is first sensed by the sensor network. The limited resources of the SNs would be finished even before starting the actual DSC. The correlation model usually does not remain constant and will change over time. The calculated correlation model is then no longer valid and must be calculated again. The need for knowing the correlation model in advance is the main disadvantage of DSC.
  - *Sensitive to environmental changes*: This is directly resulted from the first requirement of DSC to obtain and maintain an accurate correlation model. Any changes in the environment that is not

foreseen in the correlation model will lead to high errors in the collected data. Thus, DSC is not suitable for scenarios where the environment experiences very dynamic and unexpected changes.

– *Abnormal events remain undetected*: Abnormal events are unexpected events that cannot be accurately provisioned by the correlation model. They can severely impact the effectiveness of DSC by degrading the accuracy of the collected data.

2.3.3 Compressed sensing

CS theory discusses lossy compression techniques that differ from transform coding in measurement acquisition. Looking at Figure 2.3, one important observation is that a large amount of negligible compressive coefficients are discarded. CS theory tries to answer this question: Is it possible to reduce the data without performing the full transformation on the whole set of data? According to the CS theory, under certain conditions, a compressible signal can be recovered from a few number of *random linear measurements* Donoho [2006].

The aim of CS is to bring the concept behind compression techniques down to the sampling level. This is specially important in WSNs, as we prefer to compress the data as it is being transported over the network. Random measurement makes it easier to implement CS in a distributed manner, because no centralized decision and control is needed. Linearity of the measurement mechanism simplifies hardware and software design for the sensor nodes.

Similar to transform compression, CS also applies a linear transform on the sensed data. In contrast to transform compression, not all of the coefficients in the transform basis are calculated. CS transmits a fraction of the transformed data, namely *measurements*, to the sink. Acquiring linear measurements from the network involves only multiplication and additions to be calculated by the nodes and the complexity of the calculations remains to its minimum. Moreover, such operations can be performed more efficiently if a especially designed low-power hardware is built in the sensor nodes to performed array processing Bajwa et al. [2006].

The main challenge in transform coding is that, the magnitude and location of the most significant coefficients in the transformed data is not known in advance. Therefore, the transformation takes place on the whole data. CS benefits from the compressibility of the data under transformation bases, while eliminating the need to perform the transformation on the whole data. Only a subset of linear combinations of the data suffices to reconstruct the

original data. Compressibility of data under the compressive basis still plays an important role in efficiency of CS. The more compressible the raw data is, the less measurements are required to reconstruct the original data Candes [2006].

Data collection and dissemination techniques introduced in this thesis are based on the CS theory. We provide a full introduction to the CS theory and its applications in WSN in Chapter 3. To finalize our short review of CS, we summarize the pros and cons of CS as bellow.

- **Pros**:
  - *Decentralized data processing*: Most CS-based data collection designs are based on randomized measurement. A pseudo-random number generator generates a random measurement matrix that is used for calculating the measurements. The random number generator is initialized once before the operation of the WSN using the SN id as the random seed. The sink is then aware of the measurement matrix within the network without exchanging extra information, when it uses the same random number generator initialized with node id's as seed to reconstruct the random values. This mechanism avoids the communication overhead to exchange the measurement matrix between SNs and the sink.
  - *Simple arithmetic calculations at SNs*: The SNs need to calculate multiplication and additions to produce the random linear measurements. The calculation can be performed on the fly, since the coefficients are usually generated in runtime.
  - *Balanced communication and processing*: We will see in Chapter 3 that the amount of processing and communication does not increase as the measurements are being transported up the data collection tree. Unlike in-network compression, the nodes closer to the sink are not overloaded, and hence, do not become exhausted.
- **Cons**:
  - *Complex reconstruction algorithms*: After receiving the random linear measurements at the sink, the original data needs to be reconstructed from these measurements. The existing algorithms for recovery of the original data are very complex and do not scale well for very large signals such as audio or video signals. Accurate and efficient recovery algorithm work well for spatiotemporal sampling in WSN with thousands of SNs. This practically suffices for most WSN applications.

– *Inefficient measurement at leaf nodes*: Balanced communication and processing means that even leaf nodes record the same amount of measurements and send the same amount of data as the nodes that are close to the sink, although they have much less data than nodes closer to the sink. A straight-forward solution is to send raw data until the compression ratio is higher than raw data rate. We will discuss this issue and a solution for that in more details in Section 3.3.4.

2.4 Summary

In this chapter, we reviewed different approaches for data compression in WSNs. Since the focus of the thesis is on *lossy signal compression*, we focused our study on lossy compression. Lossy compression allows a better tradeoff between communication cost and accuracy of the collected data. We reviewed three major classes of lossy data compression techniques for WSNs: In-network compression using transform coding, distributed source coding, and compressed sensing also known as compressive sampling.

After giving a brief review of each technique, the advantages and disadvantages of each compression method is listed. According to the required communication cost and the provided accuracy by each method, a specific application can benefit most out of that compression method. In-network compression and transform compression are suitable for WSNs with slightly more powerful SN hardware. When combined with an efficient routing scheme, in-network compression can achieve a very high compression ratio while preserving a good accuracy of the sensed data. Distributed source coding is mainly suitable for WSNs consisting of SNs with basic hardware capabilities, because the source coding can be done with non-complex arithmetic operations. Furthermore, distributed source coding allows data collection mechanism without having memory on the SNs.

Chapter 3

Compressive Sampling in Sensor Networks

Compressive Sensing (CS) is a sampling or measurement technique to efficiently acquire compressible data without applying direct compression. CS is increasingly being applied in many areas like multimedia, machine learning, medical imaging, etc. We focus on the aspects of CS theory which has direct applications in WSNs. We also investigate the most well-known implementations of CS theory for data collection in WSNs. Before that, we have to formally define the system model that is used within the context of CS theory for WSNs.

3.1 System model

We consider the standard WSN model consisting of n static SNs distributed in an area of interest to sense a certain physical parameter of that area in a macro scale. SNs are equipped with low power and short range radio communication modules. They are either battery-powered or self-charging and possess limited processing resources. SNs sense and report environmental phenomena such as air temperature, soil humidity, light intensity, radiation level, etc. Data processing techniques for WSNs try to achieve a predefined information acquisition and preparation by exchanging local data between the SNs. Because of the short-range communication, often only neighboring nodes exchange information with each other. Therefore, node-to-node communication or node-to-sink communication may happen over multiple hops, especially when the source and destination nodes are located distantly from each other.

Throughout the thesis, the SNs are assumed to be homogeneous, unless

it is explicitly mentioned that some of the nodes may possess more computation power. Homogeneity of SNs means that all SNs possess equal hardware platforms, and therefore, have the same hardware capabilities. WSNs consisting of homogeneous SNs are particularly desired for mass deployment of SNs when no specific topology is to be enforced on the network.

We assume that a failure-free SN accurately records the physical parameter of interest at the position where it is located and the quantization error of its sensor is negligible. Sink or base station of the WSN is a high performance computation unit. It is responsible for reconstructing the state of the environment from the collected measurements. The recovered data are then prepared for use by the end user of the WSN. The WSN should be able to represent the current state of the environment periodically on a regular basis. This representation should meet a certain level of accuracy. Besides the regular monitoring, WSN should be able to detect abnormal events. An example for abnormal event is extremely high temperature at a certain location. Abnormal sensor readings serve as indicators for important events. For example high temperature recorded by a sensor node can be caused by a fire.

3.1.1 Distributed discrete signals

In sampling and signal processing literature, signal usually refers to the electrical measurements that are recorded by a sensor that samples a physical phenomenon. When several sensors record a signal at different locations, the result is a spatial signal. Sensing over time will result in temporal signals. In this text, we name the combination of spatial and temporal sampling as *spatiotemporal* sampling. Spatiotemporal sampling in WSNs refers to the signal that is recorded over consequent time instances by several SNs distributed in the environment.

Spatial signal

The data sensed by a WSN is modeled as a vector $\mathbf{f}$ of real numbers. Each item of this vector refers to a value sensed by a SN. If f_i denotes the ith component of vector $\mathbf{f}$, then f_i is equal to the value sensed by SN i. Vector $\mathbf{f}$ is often called a *discrete spatial signal* or simply signal. A WSN of size n (consisting of n SNs) senses a signal of size n, that means $\mathbf{f}$ is a real vector of length n and notated as $\mathbf{f} \in \mathbb{R}^n$. Vector $\mathbf{f}$ defines a vector in the n-dimensional real space.

Discrete temporal signal

Similar to spatial signals, the definition of the sensed data as vector can apply to the time-domain signals as well. A SN that regularly samples r times every t seconds has a temporal *sampling rate* of r/t samples per seconds. In a typical SN, the sensed data is then quantized and converted to binary data. Depending on the specific application requirements, more data processing steps may be applied before or after calculating the measurements. The discrete temporal signal recorded by the exemplified SN records a discrete signal of size r every t seconds that can be represented by a real vector $\mathbf{f} \in \mathbb{R}^r$.

Spatiotemporal signal

Spatiotemporal sampling is a direct extension of spatial and temporal sampling. For example, a WSN consisting of n SNs each of which recording r samples in every t time units, produces a discrete spatio-temporal signal that can be represented by a vector $\mathbf{f} \in \mathbb{R}^N$ where $N = nr$.

In case of sole spatial sampling, N and n are equal, because during each time period unit only one sample is recorded by a SN. Note that in parts of this text, our discussion may only focus on spatial signals without considering the temporal data. In such cases, we may use either N or n to refer to the size of the signal. According to our definition, $r = 1$ in pure spatial sampling mode, and hence, $N = n$ when no temporal sampling is taking place.

3.2 Basics of Compressive Sensing

Throughout this text we may use the terms *signal* and *vector* interchangeably. Both of them refer to discrete values of a spatiotemporal phenomenon.

3.2.1 Sparse and compressible signals

To begin formal description of the CS theory, we need some mathematical definitions. The formal definitions apply to the original signal in time or space domain as well as its projection on some frequency domain. Therefore, we use the notation $\mathbf{v}$ as a general vector which may refer to a signal or its compressive transformation. The first three definitions are general mathematical definitions which may apply to any vector. Definition 4 refers to the CS theory in a more specific manner. This clarification was required to avoid ambiguity in using the mathematical notation for the signal vectors.

Definition 1. *Vector* $\mathbf{v} \in \mathbb{R}^N$ *is said to be* S*-sparse if* $\|\mathbf{v}\|_0 = S$*, i.e.,* $\mathbf{v}$ *has only* S *nonzero entries and its all other* $N - S$ *entries are zero.*

Definition 2. *S-sparse vector* $\mathbf{v}_S \in \mathbb{R}^N$ *is made from non-sparse vector* $\mathbf{v} \in \mathbb{R}^N$ *by keeping* S *largest entries of* $\mathbf{v}$ *and zeroing its all other* $N - S$ *entries.*

Definition 3. *Vector* $\mathbf{v}$ *is said to be compressible if most of its entries are near zero. More formally,* $\mathbf{v} \in \mathbb{R}^N$ *is compressible when* $\|\mathbf{v}-\mathbf{v}_S\|_2$ *is negligible for some* $S \ll N$.

Note that $\|\cdot\|_2$ denotes the norm-2 operator[1]. The above definitions may apply to any vector which can be a signal vector or its projection on some orthonormal basis. In the coming sections, we may use the definitions 1 to 3 to both time-domain or space-domain signals or their projections on bases like Fourier, DCT, wavelet, etc. In any case, the vector $\mathbf{v}$ in the definitions above can be substituted with the signal vector $\mathbf{f}$ or its compressive projection $\mathbf{x}$ whichever is discussed in the context.

Definition 4. *Signal* $\mathbf{f}$ *is compressible under orthonormal basis* Ψ *when* $\mathbf{f} = \Psi\mathbf{x}$ *and* $\mathbf{x}$ *is compressible. The matrix* $\mathbf{\Psi}$ *is a real complex orthonormal matrix with the basis vectors of* Ψ *as its rows. We also say that* Ψ *is a compressive basis for signal* $\mathbf{f}$.

Compressibility of WSN signals

Most signals recorded from natural phenomena are compressible under Fourier, DCT and the family of wavelet transforms. This is the fundamental fact behind every traditional compression technique. Audio signals are compressible under Fourier transform. Images are compressible under DCT or wavelet. WSN also records a distributed spatiotemporal signal from a natural phenomenon which can be compactly represented under the family of Fourier or wavelet orthonormal bases.

3.2.2 Compressive measurement

CS is distinguished from traditional compression techniques in signal acquisition method as it combines compression into the data acquisition layer and tries to recover the original signal from fewest possible measurements. This is very advantageous in applications where acquiring individual samples is infeasible or too expensive. WSN is an excellent application for CS since acquiring every single sample from the whole network leads to a large traffic that can be beyond the capacity of the resource-limited SNs.

[1]For a real vector $\mathbf{v} \in \mathbb{R}^N$, norm-1 of $\mathbf{v}$ is defined as $\|\mathbf{v}\|_1 = \sum_{i=1}^{N} |v_i|$ and norm-2 of $\mathbf{v}$ is defined as $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^{N} |v_i|^2}$.

Definition 5. *Measurement matrix* Φ_m *is an* $m \times N$ *real or complex matrix consisting of* $m < N$ *basis vectors randomly selected from orthonormal measurement basis* Φ *that produces an incomplete measurement vector* $\mathbf{y} \in \mathbb{C}^m$ *such that* $\mathbf{y} = \Phi_m \mathbf{f}$.

Definition 6. *Coherence between the measurement basis* Φ *and the compressive basis* Ψ *is denoted by* $\mu(\Phi, \Psi)$ *and is equal to* $\max_{1 \leq i,j \leq N} |\langle \phi_i, \psi_j \rangle|$ *where for each* $1 \leq i, j \leq N$, ϕ_i*'s and* ψ_j*'s are basis vectors of* Φ*- and* Ψ*-domain respectively and* $\langle \cdot \rangle$ *is the inner product operation.*

3.2.3 Signal reconstruction

The novelty of the CS theory is that, it provides conditions for accurate recovery of the original data by applying a polynomial-time reconstruction algorithm on the collected compressive measurements.

Theorem 1. *Suppose signal* $\mathbf{f} \in \mathbb{R}^N$ *is* S*-sparse in* Ψ*-domain, i.e.,* $\mathbf{f} = \Psi \mathbf{x}$ *and* $\mathbf{x}$ *is* S*-sparse. We acquire* m *linear random measurements by randomly selecting* m *basis vectors of the measurement basis* Φ. *Assume that* $\mathbf{y} \in \mathbb{C}^m$ *represents these incomplete measurements such that* $\mathbf{y} = \Phi_m \mathbf{f}$ *where* Φ_m *is the measurement matrix. Then it is possible to recover* $\mathbf{f}$ *exactly from* $\mathbf{y}$ *by solving the following convex optimization problem:*

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in \mathbb{C}^N}{\operatorname{argmin}} \|\mathbf{x}\|_1 \quad \textit{subject to} \quad y_k = \langle \Psi \mathbf{x}, \phi_k \rangle \quad \textit{for all} \quad k = 1, \ldots, m. \tag{3.1}$$

where ϕ_k*'s are the rows of the* Φ_m *matrix. Recovered signal will be* $\hat{\mathbf{f}} = \Psi \hat{\mathbf{x}}$ *Candes and Tao [2006].*

In case of real measurement and compressive bases, problem (3.1) can be simplified to a linear program Vanderbei [2001]. Noiselet measurement matrices involve complex numbers and thus a linear program can not solve problem (3.1) Coifman et al. [2001]. Convex optimization problem (3.1) with complex values can be cast to a Second Order Cone Program (SOCP) Winter et al. [2005].

Sparsity and incoherence

Accurate signal recovery is possible when the number of measurements follows

$$m > C \cdot S \cdot \log N \cdot \mu^2(\Phi, \Psi) \tag{3.2}$$

where $C > 1$ is a small real constant Candes and Romberg [2007].

From Equation (3.2) it is clear why sparsity and incoherence is important in CS. In order to efficiently incorporate CS theory in a specific sampling scenario, we need the measurement and compressive bases to be incoherent as maximum as possible to decrease parameter μ in (3.2). Moreover compressive basis must be able to effectively compress the signal $\mathbf{f}$ to decrease S in (3.2). When these two preconditions hold for a certain sampling configuration, it is possible to recover the signal $\mathbf{f}$ from m measurements where m can be much smaller than the dimension of the original signal Candes and Tao [2006].

Random measurements

Interestingly, random matrices such as a Gaussian matrix with independent and identically distributed (i.i.d.) entries from a $\mathcal{N}(0, 1)$ have low coherence with any fixed orthonormal basis Candes and Romberg [2007]. The elements of such a random matrix can be calculated on the fly using a pseudo-random number generator which is common between SNs and the sink. When the normal random generator at every SN is initialized by the id-number of that SN, the sink can also reproduce the measurement matrix. Note that in this case, there is no need for a centralized control to update the measurement matrix and the values of the measurement matrix are not needed to be stored inside the SN. Therefore, using random measurement matrices gives us more flexibility and requires less memory on the SNs. Instead, because of slightly more coherence between random measurement matrix and the fixed compressive basis, the number of required measurements m will increase according to (3.2).

Noisy measurements

CS is also very stable against noisy measurement and can also handle signals that are not strictly sparse but compressible. It is a very idealistic condition being able to transform signal $\mathbf{f}$ to a strictly sparse vector in Ψ-domain. Instead, $\mathbf{f}$ is usually transformed into a compressible form with many near zero entries. Candes et al. have shown that if $\|\mathbf{x} - \mathbf{x}_S\|_2 < \epsilon$ for some integer $S < N$ and a small real constant ϵ, then the recovery error by solving problem (3.1) will be in order of $O(\epsilon)$. Similarly, in a noisy environment if the measurement vector is added by an Additive White Gaussian Noise (AWGN) $\sim \mathcal{N}(0, \sigma^2)$, the recovery error is bounded by $O(\sigma^2)$ Candes et al. [2006a,b].

3.3 Compressive Sensing for WSNs

In this section, we review the major adaptations of the CS theory for a distributed implementation in WSNs.

3.3.1 Compressive Wireless Sensing

Bajwa et al. Bajwa et al. [2006] first proposed an implementation of CS for WSNs which was based on analog data communication. Although the realization of their abstract idea is merely feasible, their model provides a simple technique to acquire compressive measurements from a WSN. Therefore, we begin the study of major CS implementations for WSNs first by introducing Compressive Wireless Sensing (CWS).

Calculations at the sensor nodes

Let Φ be a measurement basis and $\mathbf{\Phi}_m$ is a measurement matrix that has m rows and N columns. Here N is the total number of SNs and m is the required number of compressive measurements. m can be determined by Equation (3.2) according to the compressibility level of the spatial signal. The measurement matrix $\mathbf{\Phi}_m$ is made of m randomly selected basis vectors from the measurement basis Φ. When we are using random Gaussian basis, $\mathbf{\Phi}_m$ can be simply populated by random real numbers from a Gaussian distribution with zero mean and unit variance. Either using random measurements calculated on the fly, or using embedded measurement vectors, we have to calculate the result of $\mathbf{\Phi}_m\mathbf{f}$. Here $\mathbf{f} \in \mathbb{R}^N$ is the spatial signal with a dimension equal to the number of SNs. The matrix multiplication $\mathbf{y} = \mathbf{\Phi}_m\mathbf{f}$ is actually a linear measurement process which must be performed in a distributed manner. The result will be the vector $\mathbf{y} \in \mathbb{R}^m$ which has lower dimension that the original signal. The CS recovery algorithm then has to recover the original vector $\mathbf{f}$ from $\mathbf{y}$.

Accumulation through network coding

To understand how CWS works, we consider the simplest case where all SNs can directly communicate with the sink. We know that $\mathbf{\Phi}_m$ has N columns. Suppose each SN embeds one column of $\mathbf{\Phi}$. When using random measurements, no embedded data is required. The ith column of $\mathbf{\Phi}_m$ can be generated on the fly. The only requirement here is that the entries of $\mathbf{\Phi}_m$ can be regenerated at the sink. Therefore, either the columns of $\mathbf{\Phi}$ are embedded into SNs before deployment of the WSN, or SNs produce their corresponding

column using a pseudo-random Gaussian random number generator which is known to the sink. The same $\mathbf{\Phi}_m$ can be then reproduced at the sink using the same algorithm and a predetermined seed. When the SNs initialize the pseudo-random Gaussian random number generator with their own id, the whole sequences of random entries of the measurement matrix can be regenerated at each sampling period.

All SNs record the value of the intended physical phenomena at the same time. This means that the nodes have synchronized clocks (we will discuss later the disadvantages of the need for synchronized clocks). Each SN then multiplies its recorded real value with its own column vector which is part of the measurement matrix $\mathbf{\Phi}_m$. We set $\mathbf{w}_i = f_i\phi_i$ where f_i is the value recorded by that SN and ϕ_i is the ith vector of $\mathbf{\Phi}_m$. When all SNs have done the calculation of their own vector multiplication, the vector $\mathbf{w}_i$ will be ready to be transmitted from each SN. If all the SNs transmit their own $\mathbf{w}_i$ at the same time, the result of matrix multiplication $\mathbf{y} = \mathbf{\Phi}_m\mathbf{f}$ will be accumulated at the sink. Figure 3.1 illustrates this setup. CWS assumes that the SNs can be perfectly synchronized such that the measurement vector $\mathbf{y}$ can be accurately accumulated at the sink.

Extension to tree topology

The accumulation is actually taking place by adding vectors of the same size across the WSN. Therefore, it can be done by any order since the addition is a commutative operation. This property of CS measurement has a very important advantage. Due to the commutative property of addition, accumulation can be done in any order. Therefore, the same approach as discussed above can be applied to a star topology, chain topology or more complex tree topologies. As long as the tree routing is applied, accumulation can be done over many levels. Thus, we can extend the very simple topology in Figure 3.1 to a multi-hop WSN depicted in 3.2.

Limitation of CWS

The CWS model in its very primary form which is based on synchronized superposition of signals is hard to implement with current hardware of today's typical SNs. However it has interesting attributes in its abstract form. Note that the accumulation of the vectors can be done on the air and without further computation by the SNs. In the star topology the accumulation takes place instantly when all SNs have transmitted their own calculated vector. Today's common wireless Medium Access Control (MAC) protocols for WSNs are all based on detecting and avoiding collision. Therefore, special

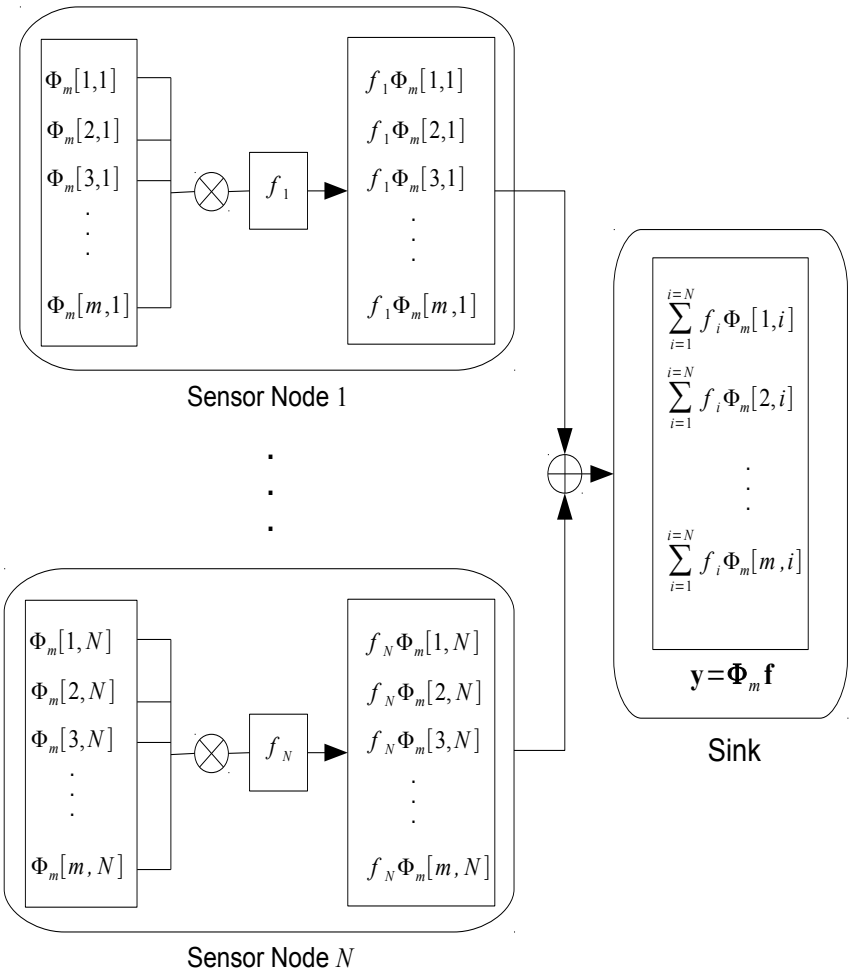


Figure 3.1: CWS in a WSN with star topology

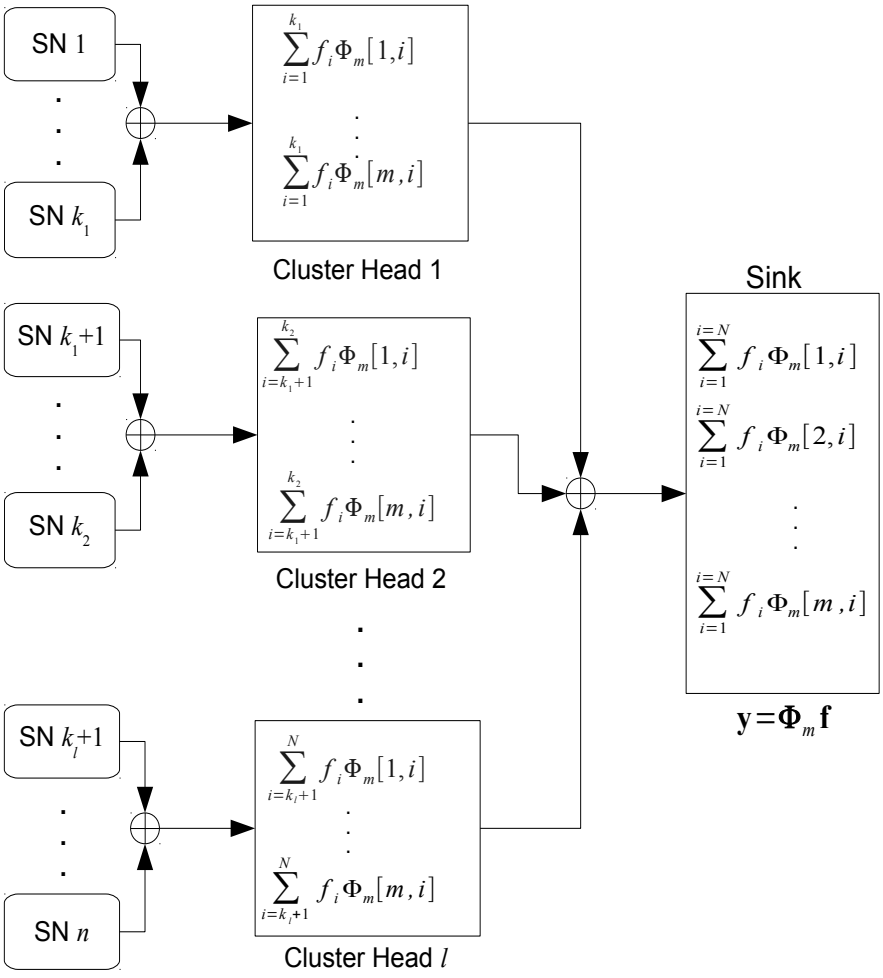


Figure 3.2: CWS in a multi-hop WSN with tree topology

hardware and radio modules must be developed that allows signal collision and superposition to realize an implementation of CWS.

There has been a large body of continuous work for an efficient MAC protocol that faces minimum collisions. In contrast to all these achievements, CWS requires a synchronized collision. Implementing a communication layer with perfectly synchronized collisions is even more challenging than making a wireless MAC protocol with minimum collisions. This topic can be considered as an open area in applying CS theory in WSNs.

3.3.2 Compressive Data Gathering

A more practical implementation of the CWS concept is introduced by Luo et al. in Compressive Data Gathering (CDG) for large-scale WSNs Luo et al. [2010, 2009]. Using overcomplete dictionaries Donoho et al. [2006] in solving (3.1), CDG is also able to tackle abnormal sensor readings and cope with unpredicted events in the operational environment of WSN. CDG models the abstract data aggregation discussed in CWS as message passing through the WSN. Therefore, we do not discuss the data transfer as analog signals any more. Instead, data transmission is modeled as messages that are transferred over a digital wireless communication channel. CDG in its heart remains very much like CWS, while paying attention to implementation details and practicality of its model.

Effect of abnormal sensor readings

Figure 3.3 shows an example of the effect of the abnormal samples on the compressibility of an illustrative signal. Note that the signal is synthetically generated and is not based on real data. Therefore, it does not contain a unit for its values. The Y-axis is unit-less and the X axis indicates the sample index in signal vector or coefficient index in its transformed vector. Figure 3.3(a) shows the original signal which is sparse in DCT as can be seen in Figure 3.3(b). In Figure 3.3(c) we see the same signal with two abnormal sensor readings. The values of those two samples are either too large or too low compared to the normal sample values. Figure 3.3(d) shows the DCT transform of the abnormal signal. Obviously the abnormal signal is not compressible under DCT. This violates all of the preconditions required for operation of the traditional CS. Fortunately, there is a solution to this problem by using over-complete dictionaries Donoho et al. [2006].

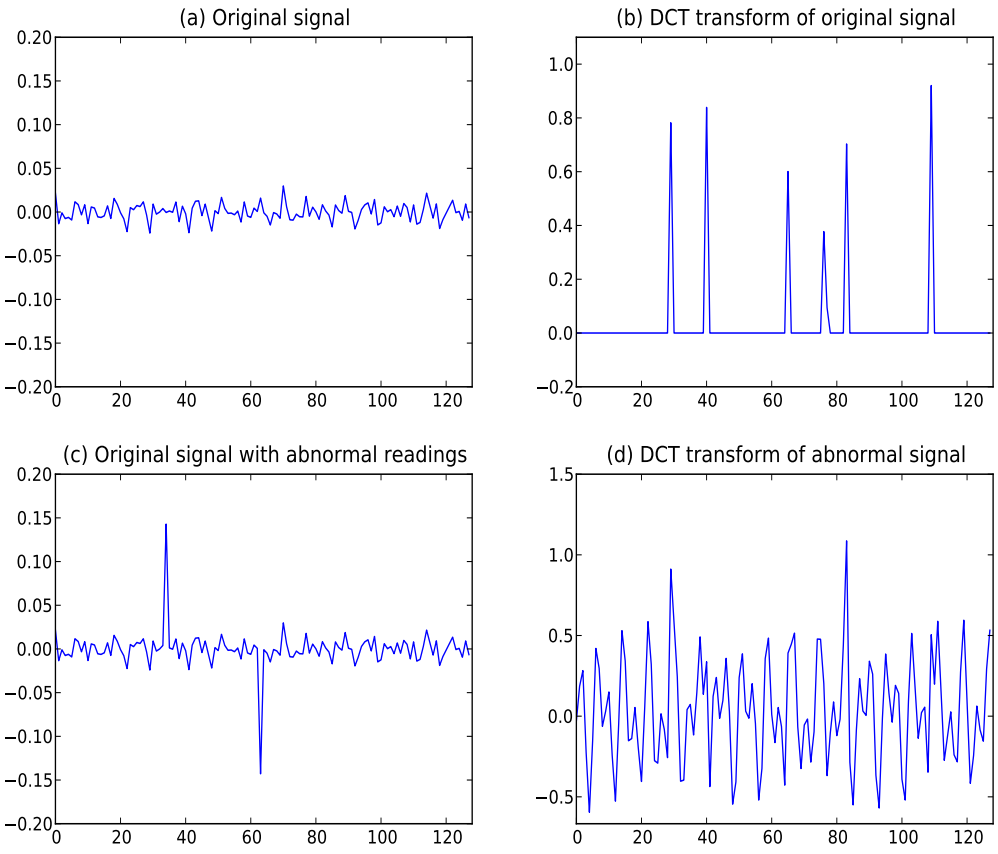


Figure 3.3: Effect of abnormal readings on the DCT projection of an illustrative signal (magnitude applies to any physical unit)

Over-complete compressive basis

CDG proposes the use of over-complete dictionaries to detect abnormal sensor readings. Assume $\mathbf{f}$ is the spatial signal vector that may have been contaminated with abnormal readings. Remember that we assume that the abnormal readings in the signal are sparse, otherwise it is not possible to recover a heavily distorted signal by any efficient data acquisition mechanism. We can decompose the vector $\mathbf{f}$ into two vectors:

$$\mathbf{f} = \mathbf{f}_0 + \mathbf{d}_s \tag{3.3}$$

where $\mathbf{f}_0$ is the signal without abnormal sensor readings which is sparse or compressible in the compressive basis Ψ and $\mathbf{d}_s$ contains the deviated values of the abnormal readings. The vector $\mathbf{d}_s$ is supposed to be sparse since the abnormal sensor readings are rare and sporadic. Therefore, $\mathbf{d}_s$ is sparse in space domain. The original signal vector $\mathbf{f}$ can be then represented as the linear combination of two sparse signals and we can express the signal acquisition by:

$$\mathbf{f} = \mathbf{\Psi}\mathbf{x}_0 + I\mathbf{x}_s \tag{3.4}$$

where $\mathbf{x}_0$ is the sparse projection of the normal component $\mathbf{f}_0$ on the Ψ-domain. Vector $\mathbf{d}_s$ is in fact equal to $\mathbf{x}_s$ since $\mathbf{d}_s$ is already sparse in the space domain. I is the identity matrix, and hence, $\mathbf{d}_s = \mathbf{x}_s = I\mathbf{x}_s$. It is possible to project the signal $\mathbf{f}$ on a single over-complete basis. First we need to construct an augmented transformation matrix into the over-complete system named Ψ'. Assume that the matrix $\mathbf{\Psi}'$ is made by putting the matrices $\mathbf{\Psi}$ and I next to each other. Formally speaking, $\mathbf{\Psi}' = [\mathbf{\Psi} I]$ which has now N rows and $2N$ columns.

Donoho et al. have shown the possibility of stable recovery of sparse signals under an over-complete system Donoho et al. [2006]. They have also proved that their method is effective for measurements contaminated with noise. The recovery takes place using a convex optimization similar to the typical CS recovery method. The recovery error in presence of noise would be in the order of the additive noise magnitude. Since we are going to search for a solution in an augmented dictionary, our target vector has now the dimension of $2N$ instead of N. Note that $\mathbf{f} \in \mathbb{R}^N$ can be represented by a sparse projection on the Ψ' basis:

$$\mathbf{f} = \mathbf{\Psi}'\mathbf{x} \quad , \quad \mathbf{x} = [\mathbf{x}_0{}^T \mathbf{x}_s{}^T]^T \tag{3.5}$$

Here you see that $\mathbf{x}$ has a dimension of $2N$ and is made by concatenating $\mathbf{x}_s$ and $\mathbf{x}_0$ on top of each other.

Detecting abnormal sensor readings

Now we apply CS reconstruction as explained in Theorem 1 for this new pair of signals and measurements. The measurement vector can be calculated using a random measurement matrix as before. The estimated solution vector will be $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_0^T \hat{\mathbf{x}}_s^T]^T$. The original signal can be estimated by calculating $\hat{\mathbf{f}_0} = \mathbf{\Psi}\hat{\mathbf{x}_0}$. Again the recovery error depends on the number of measurements m and the noise which is present in the communication channel. CDG is also able to detect the location of abnormal events in the WSN. After solving the convex optimization problem (3.1) for the augmented system and finding the solution $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_0^T \hat{\mathbf{x}}_s^T]^T$. The significant nonzero elements in $\hat{\mathbf{x}_s}$ determine the position of the events.

3.3.3 Distributed Compressed Sensing

Wakin et al. in Distributed Compressed Sensing (DCS) give a model of WSN with each node having direct link to the sink Wakin et al. [2005]. DCS considers not only the spatial correlation of a distributed signal but also the temporal correlation over a period of time. Therefore, DCS can be regarded as a spatiotemporal sampling technique. The spatiotemporal signal is modeled as combination of several temporal signals. DCS assumes two components are contributing to the spatiotemporal signal recorded by the WSN. The first component is the common component which is sensed overall by SNs. For the second component individual SNs can contribute their own signals which can be sparse. According to this configuration, DCS defines three Joint Sparsity Models (JSMs) namely JSM-1, JSM-2 and JSM-3 which will be briefly explained later. Suppose there are J SNs each of which are producing a temporal signal $\mathbf{f}_j$ with dimension N. This means J SNs are producing N samples in every sampling period. We assume that there is a fixed Ψ-basis for $\mathbb{R}^N$ on which the signals can be projected sparsely. Now, we explain attributes of the three JSM models.

JSM-1: Sparse common component + innovations

In this model, all of the temporal signals share a common component which is compressible in Ψ-domain while every SN may contribute a sparse innovation component:

$$\mathbf{f}_j = \mathbf{z}_C + \mathbf{z}_j \ , \ j \in 1, 2, ..., J \tag{3.6}$$

such that

$$\mathbf{z}_C = \mathbf{\Psi}\theta_C \quad , \quad \|\theta_C\|_0 = K \qquad \text{and} \qquad \mathbf{z}_j = \mathbf{\Psi}\|\theta_j\|_0 \quad , \quad \|\theta_j\|_0 = K_j \tag{3.7}$$

Note that the operator $\|\cdot\|_0$ denotes the number of nonzero elements in it operand vector. In the above formula, $\mathbf{z}_C$ is the common component which has the K-sparse representation of θ_C. Every SN has its own innovation $\mathbf{z}_j$ which is K_j-sparse in Ψ-domain. Note that here $\mathbf{\Psi}$ is an $N \times N$ orthonormal matrix. This model is probably the most realizable WSNs. Imagine a WSN that is deployed in an outdoor environment to record the micro-climate data such as ambient air temperature. Overall, nearly a constant temperature is sensed which is close to the average. There might be some more structures present in the signal values over a geographical area, but the signal is expected to be temporally compressible. In some spots of the operational environment, there is a temperature difference because of local events or conditions like shades, flow of water, etc. Therefore, the JSM-1 model is suitable for modeling this signal.

JSM-2: Common sparse supports

In this model, signals recorded by all SNs can be projected sparsely on a single fixed orthonormal basis, but the coefficients may differ for every SN. Formally speaking this means that

$$\mathbf{f}_j = \mathbf{\Psi}\theta_j \quad , \qquad j \in 1, 2, 3, ..., J \tag{3.8}$$

where each signal vector θ_j can be projected only on the same subset of basis vectors $\Omega \subset 1, 2, 3, ..., N$ such that $|\Omega| = K$. Therefore, all temporal signals recorded by SNs, have a sparsity of degree K, but the amplitude of coefficients may differ for each SN. This model is useful in situations such as in acoustic sensor networks where all signals have the same support in frequency domain, but may experience phase shifts and attenuations due to the signal propagation.

JSM-3: Non-sparse common + sparse innovations

This model can be regarded as an extension to JSM-1 since it does not require the common signal to be sparse in the Ψ-domain, however the innovations by individual SNs is still sparse:

$$\mathbf{f}_j = \mathbf{z}_C + \mathbf{z}_j \quad , \qquad j \in 1, 2, ...J \tag{3.9}$$

$$\mathbf{z}_C = \mathbf{\Psi}\theta_C \qquad \text{and} \qquad \mathbf{z}_j = \mathbf{\Psi}\theta_j \quad , \quad \|\theta\|_0 = K_j. \tag{3.10}$$

One application of this model can be in a distributed deployment of cameras. The overall picture can be presented as a reference picture with some small

innovations sensed by different cameras. These differences from the base picture depend on the position of each camera. Wakin et al. has also proposed a new algorithm especially developed for joint signal recovery.

3.3.4 Compressive Sensing over ZigBee Networks

ZigBee is a high level wireless communication specification based on IEEE 802.15.4 standard Caione et al. [2010]. It is widely used in today's SN platforms. As we mentioned before, the CS variants designed for WSN require further implementation adaptations to become suitable for the typical hardware and software platform that is currently being used. CWS requires analog signal superposition, and hence, perfect time synchronization between nodes. Furthermore, it does not yet consider the effect of multi-path fading and other propagation noises. CDG follows a more practical approach, though it does not target any standard and established communication protocol or SN hardware platform. DCS and JSM models are best suited for star topologies where each node can transmit its data directly to the sink. Similar to CWS, a time synchronized signal superposition is preferred for DCS, otherwise Time Division Multiple Access (TDMA) methods will become very time consuming.

Raw sampling at leaf nodes

Caione et al. proposed a simple and yet very effective improvement technique for the operation of a CS-based signal acquisition for WSNs Caione et al. [2010]. In the previous section, we have seen that when using CS, the number of data items to be transmitted by each SN is equal to m and hence the energy consumption is balanced. This means that for a leaf node that needs to send only one value, m transmissions must be performed. We know that most of the nodes in a tree structure belong to the lower levels of the tree (levels near to the leaves). Therefore, a majority of nodes are sending data much more than the useful information that they produce. Caione et al. introduced a *Hybrid CS* method for WSNs in which two operations are performed by the SNs:

- Pack and Forward (PF): the SN packs its own recorded data along with the data received from its children SNs.
- CS measurement: the SN accumulates its data and the data that has been received from its children using CS measurement techniques. It then sends the CS measurement vector to its parent node.

All SNs follow two rules throughout acquiring, packing, forward and accumulation of data:

- A SN only applies CS measurement accumulation method, when the length of the resulting message is less that when using PF. In other words, each node decides whether to use PF or CS depending on the length of the outgoing message. This way the SNs minimize the energy consumed during radio transmission.
- When the message received by a SN is a CS measurement, the SN is obligated to apply the CS accumulation process. In other words, when the message changes its type from PF to CS, then it continues to be in CS form all the way up through the network till it reaches the sink.

These two simple rules and operations help to reduce the overall energy consumption of SNs. However, the energy consumption is not balanced any more. One drawback of the discussed hybrid CS scheme is that there is still the possibility of network partitioning, since the nodes near to the sink may get exhausted and deplete earlier than leaf SNs.

3.4 Summary

In this chapter we have explained the fundamentals of the emerging theory of CS and its applications in WSNs. We have realized that CS provides a very flexible, tunable, resilient and yet efficient sampling method for WSNs. We have listed major advantages of CS over transform coding or other signal compression methods which. CS guarantees a balanced energy consumption by all of the SNs. This important property avoids network partitioning and improves overall resource management of the whole network. It is easily implementable on strict hardware or software platforms of today's typical SNs. Its resilience against noise and packet loss, makes it very suitable for the harsh operational environment of WSNs.

We have reviewed most important variants of CS which are especially devised for WSNs. In recent years, with the progress of the CS theory in WSNs area, more and more realistic applications of CS are presented. CWS was one of the first variants of CS for WSNs. Though it lacks consideration of hardware implementability, it offers a novel approach which openes a new avenue of research in the area of signal acquisition for WSNs. CDG which is directly based on CWS model, enabled detection of unexpected events. Event detection is a crucial requirement of many critical WSN applications.

DCS with its JSM models has categorized a wide range of applications for CS in WSNs. DCS provided an efficient solution to the problem of joint

signal recovery. DCS has many applications in different distributed sampling scenarios, like WSN with star topology, camera arrays, acoustic localization, etc. However, it is not very suitable for multi-hop WSNs. Hybrid CS is a simple and yet effective solution for multi-hop WSNs consisting of very resource-limited SNs. It is one of the most recent improvements to typical CS implementation in WSNs. The only disadvantage of using Hybrid CS is the unbalanced energy consumption that may lead to network partitioning.

Chapter 4

Reordering for Better Compressibility

In this chapter, we introduce our enhancement to CS in WSN by finding a better labeling (indexing) of the SNs. Our improvement does not affect the basics of CS or its WSN implementation. Consequently, the state of the art techniques can be combined with the methods introduced here. We show that if the signal vector is viewed under a reordering (mapping) function, it is possible to obtain a more compressible signal, i.e., a signal which is sparser in a certain domain such as DCT. Our work offers a new approach to the WSN sampling problem by enhancing the performance of CS.

The major advantages of our proposed technique are:

- A polynomial-time algorithm that finds a permutation of samples of the discrete signal vector $\mathbf{f}$, so that the linear transform of $\mathbf{f}$ in frequency domain is sparser than the original ordering of samples.

- A CWS model which is capable of adapting itself to the environment changes. When the state of the environment does not change quickly, our model is capable of reducing spatial sampling rate through constructing a more compressible view of the spatial signal.

In addition to reconstructing the original signal from compressively sampled data, the sink has a second responsibility in our enhanced compressive sampling scheme: Computing sub-optimal reordering of the SNs with sparser DCT representation. We assume the sink to be powerful in processing, i.e., computation and storage.

4.1 Motivation for reordering

In most distributed CS applications, sampling and sparse bases are determined prior to the deployment of the sensor network. Therefore, according to Equation 3.2, μ remains constant because compressive and measurement domains, namely the bases Φ and Ψ do not change after network deployment. Therefore, sparsity factor S is the only parameter that is effective on the minimum number of required measurements m. If we succeed to decrease S, then we can reconstruct the original signal from fewer samples, saving valuable bandwidth and SNs energy.

4.1.1 Conventional indexing of SNs

In some sampling problems such as WSNs, the ordering of the samples is not dictated by an independent phenomenon. Mostly, we can assign the value sensed by one sensor to the first signal element and the other to the second one, and so on. In this chapter, we focuses on such conditions where we can set the sampling order.

It is important to emphasize that the position and order of SNs are different aspects. Reordering only takes place at the sink and all its required computations are done outside the WSN. That's why our proposed model does not add overhead to the WSN nodes. In simple words, reordering is an alternative view of our WSN signal vector under which we can apply CS more efficiently. It does not require to relocate the SNs or change their positions.

Less measurement requirements after proper reordering

Our enhancement works by improving compressibility, that means increasing sparsity of $\mathbf{f}$ under the Ψ-transform, which means decreasing S in Equation 3.2. Then, from Equation 3.2 it implies that a view of the signal that makes it appear to be more compressible, leads to fewer number of compressive measurements required for signal reconstruction. Lower compressive sampling rate means more efficient bandwidth usage and decreased energy consumption, and hence, prolonging WSN lifetime.

Energy consumption by the SNs is directly related to data transmission rate and number of compressive measurements, but it is inversely related to compressibility of the signal that is sensed by the WSN. Compressibility is increased by finding a mapping under which the WSN signal $\mathbf{f}$ is sparser in Ψ-domain.

4.2 Problem formulation

WSN can operate in two modes that we name *full measurement mode* or simply full mode and *sub-measurement mode*. Let S_e be the expected value for the sparsity or compressibility of the spatial signal without applying the reordering of the SNs. In full mode, from our previous knowledge or estimation of S_e in the operational environment, we determine the expected number of measurements m_e by substituting S by S_e in Equation 3.2.

First m_e measurements are acquired. Measurement acquisition takes place by any suitable implementation of CS for WSN, such as CWS, CDG and other techniques introduced in Chapter 3. In sub-measurement mode, we calculate a suboptimal reordering of the signal at the sink. Then the sink broadcasts a new version of the sampling matrix $\mathbf{\Phi}_{m\times n}$ with fewer rows, as the minimum required samples (m) is reduced because signal's sparsity is improved through our reordering.

Without loosing generality, we limit our configuration to the case where $\mathbf{\Phi}$ is an orthogonal Gaussian random basis and Ψ to be the Fourier domain. Our conclusions will also apply to any other pair of orthogonal incoherent sampling and sparse bases. One can apply this method to other bases such DCT or wavelets. In fact, the method discussed here applies to any other transformation (projection) bases which can be presented in matrix multiplication form. For example one can set $\mathbf{\Psi}$ to be the DCT matrix, and find the sub-optimal reordering as described by our method detailed in this section.

The elements (samples) of signal $\mathbf{f}$ are initially ordered by SNs' ids. Our objective is to find a permutation under which the Ψ-transform of $\mathbf{f}$ is sparser.

4.2.1 Combinatorial problem statement

A permuted function π is defined as a one-to-one mapping from a set of natural numbers to a permuted set of the same numbers. Here, we consider the case where the set $\{1, 2, 3, \ldots, n\}$ is mapped to to a permutation of the same elements.

$$\begin{aligned} &\pi : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\} \\ &\forall i, j \in \{1, 2, \ldots, n\} : i = j \Rightarrow \pi(i) = \pi(j) \\ &\forall i \in 1, 2, \ldots, n : \exists j \in \{1, 2, \ldots, n\}\ \pi(j) = \pi(i) \end{aligned} \tag{4.1}$$

Suppose $\mathbf{f}_\pi$ is a reordered version of signal $\mathbf{f}$ according to the permutation function π. That means, $\mathbf{f}_\pi$ has the same samples as $\mathbf{f}$, though in a different

order.

$$\forall i \in \{1, 2, \ldots, n\} \quad \mathbf{f}_\pi[i] = \mathbf{f}[\pi(i)] \tag{4.2}$$

Let Π be the set of all possible permutation functions defined on set $\{1, 2, \ldots, n\}$. Apparently, the size of set Π is $n!$, i.e., $|\Pi| = n!$. The optimal permutation function for signal $\mathbf{f}$ is the permutation function π^* that gives the sparsest vector under a Ψ-transform.

$$\forall \pi \in \Pi \;\; \pi \neq \pi^* \quad \mathbf{\Psi f}_{\pi^*} \text{ is sparser than } \mathbf{\Psi f}_\pi \tag{4.3}$$

4.2.2 Condensing the energy of the signal

Fourier domain.

The inverse projection of the unit vectors in the Fourier domain to the time/space domain, gives us some vectors in the sampling domain. If we find an order of the samples that best matches one of these basis vectors, we can expect that the Discrete Fourier Transform (DFT) or DCT of the optimally reordered sample vector has most of its energy concentrated on a few Fourier coefficients. If we set $\mathbf{\Psi}$ to be the DFT or DCT matrix, then the inverse projection basis vectors in the sampling domain will be the columns of $\mathbf{\Psi}^{-1}$ (this is also valid for any other transform which can be represented as matrix multiplication of an orthonormal transformation matrix). For each basis vector, we can find the optimal permutation of the samples, resulting in n different permutation functions for each of the n basis vectors. Among these we choose the permutation that matches a basis vector better than the others.

In our model, $\mathbf{f}$ is the discrete spatial vector whose each element is the real value sensed by corresponding SN. Initially, the elements of $\mathbf{f}$ are ordered simply by SNs id's. Our objective is to find a permutation π^* under which Ψ-transform of $\mathbf{f}$ is sparser. Now for each basis vector $\mathbf{f}$ we construct a *difference matrix* $\mathbf{\Delta}$ as below:

$$\mathbf{\Delta}_{n\times n} = \begin{pmatrix} |f_1 - \psi_1| & |f_1 - \psi_2| & \cdots & |f_1 - \psi_n| \\ |f_2 - \psi_1| & |f_2 - \psi_2| & \cdots & |f_2 - \psi_n| \\ \vdots & \ddots & & \vdots \\ |f_n - \psi_1| & |f_n - \psi_2| & \cdots & |f_n - \psi_n| \end{pmatrix} \tag{4.4}$$

We define two vectors with least difference as most matching. According to this definition, finding π^* in Equation 4.3 is equivalent to finding n elements of $\mathbf{\Delta}$ with minimum sum, such that no two of them are on the same row or column. Note that, we consider only near-optimality of the

permutation and do not claim finding the globally optimal solution. As our evaluations show, our *nearly-optimal* reordering also leads to a significantly more compressible signal.

4.3 Reordering for enhanced CS in WSNs

In the following, we introduce a simple greedy algorithm that finds a permutation of the n elements of $\mathbf{f}$ in $O(n^2 \log n)$ time. Next, we show how to use our algorithm to improve compressibility, and hence, the performance of CS in WSNs.

4.3.1 Greedy approximate solution

Here we introduce our algorithm (Algorithm 1) for finding a sub-optimal permutation of the samples, called the Sub-Optimal Permutation (SOPerm). Note that this algorithm only compares two unit vectors (or two vectors of the same scale) and outputs a sub-optimal permutation to project the first vector on the second. Here ψ's are the unit vectors of the sparse representation domain and $\mathbf{f}$ is our discrete signal vector. The basis vectors (ψ's) are fixed and we have to find a permutation for $\mathbf{f}$ which is sub-optimal.

Note that Algorithm 1 requires the vectors $\mathbf{f}$ and ψ to be normalized in order to have the same scale. Normalization is necessary to eliminate the effects of signal magnitude on the differences between signal elements and basis vector components. Obviously, an optimal permutation found for normalized versions of $\mathbf{f}$ and ψ is also optimal for the original vectors $\mathbf{f}$ and ψ.

1: **for** $i = 1$ .. n **do**
2: **for** $j = 1$ .. n **do**
3: $\Delta_{i,j} \leftarrow |f_i - \psi_j|$
4: **end for**
5: **end for**
6: $C \leftarrow \varnothing$, $R \leftarrow \varnothing$, $\sigma \leftarrow 0$
7: **for** $k = 1$ .. n **do**
8: Select $i \in \{1, 2, \ldots, n\} - R$ and
$j \in \{1, 2, \ldots, n\} - C$, so that
$\Delta_{i,j}$ is minimum among all i and j
9: $R \leftarrow R \cup \{i\}$, $C \leftarrow C \cup \{j\}$, $\sigma \leftarrow \sigma + \Delta_{i,j}$
10: $\pi_j = i$
11: **end for**

Algorithm 1: Sub-Optimal Permutation (SOPerm)

SOPerm first constructs the matrix $\mathbf{\Delta}$. Then, it runs n steps and at each step greedily selects the element which is not on a row or column of a previously selected element and the addition of that element to the sum of previously selected elements is minimum. This is equal to adding the smallest non-constrained element of $\mathbf{\Delta}$, because all the elements of $\mathbf{\Delta}$ are positive real numbers.

In Algorithm 1, Lines 1-5 constructs the matrix $\mathbf{\Delta}$. Line 6 initializes the set of visited rows and columns (R and C respectively) to null. Our greedy approach is implemented in Lines 7-11. At each iteration of the for-loop, an element at the i^{th} row and j^{th} column of $\mathbf{\Delta}$ with minimum value is selected, such that i does not exist in R and j does not exist in C. At Line 9 inside the for-loop, the i^{th} row and the j^{th} column are marked as visited by appending them to the sets R and C respectively.

Because the spatial DFT/DCT of a natural environment is expected to have only low frequencies, we begin our search for the sub-optimal permutation from low frequency basis vectors. Our simulations justify this hypothesis, as searching in higher frequencies resulted no better permutations than the one found among low frequencies. Therefore, to decrease processing time our SOPerm algorithm is run against lower frequency basis vectors.

4.4 Application of reordering in CS-based WSNs

So far we have presented a practical method to redefine a model of the WSN under which we can improve signal sparsity, and hence, decrease compressive sampling rate. The SOPerm algorithm will be implemented on the sink (outside the WSN). SOPerm tries to find a reordering of SNs that leads to a sparser discrete spatial signal. Consider that the spatial CS takes place on a regular basis every T time units (selecting a proper T that is suitable for the dynamics of the signal and characteristics of the operational environment is another important problem that is out of scope of this text).

4.4.1 Adapting the permutation

At the beginning, we set the minimum required samples for reconstruction, to be m_e calculated according to Equation 3.2 with presumption of signal $\mathbf{f}$ being S_e-sparse. SOPerm is then run for the reconstructed signal $\mathbf{f}^*$ which is expected to be almost exactly equal to $\mathbf{f}$.

SOPerm calculates a sub-optimal permutation π^* under which vector $\mathbf{f}$ is S-sparse and $S < S_e$. For the next sampling round, we use this permutation

and sample at a rate of at least $m = C{\cdot}\mu^2(\Phi, \Psi){\cdot}S{\cdot}\log n < m_e$. Given that π is still a good reordering for the next T time units because the environment does not change so quickly over time. We emphasize again that our model applies only to WSNs with fixed SNs in an environment with moderate changes over time. Our reordering solution assumes that drastic perturbations in the signal value do not occur. We will discuss scenarios with more perturbations especially in Chapters 5 and 6.

4.4.2 Reusing the permutations over multiple sampling rounds

By taking slightly more measurements, i. e., $m + r$ measurements such that $r \ll m$, more certainty can be gained that we acquire enough compressive samples to reconstruct the original signal. In the subsequent sampling rounds and using our new reordering, we can reconstruct the spatial signal $\mathbf{f}$ from fewer measurements. The whole process is repeated for the next sampling round. The amount of data is then reduced because the new permutation of the samples has more compressibility.

Figure 4.1 illustrates the overall architecture of our adaptive reordering model. Our model is in fact a closed-loop system that applies CWS every T time units and updates its internal model according to the changes in the operational environment. Because m changes over time, the sampling matrix Φ has to be broadcast to all SNs before each sampling round. As an alternative, one can find a way to publish only the integer number m and then SNs shall compute their own individual columns of Φ according to a predetermined seed for generating a pseudo-random sampling matrix. The second method requires much less broadcasting and is particularly suitable for WSNs with randomized measurement matrices.

4.4.3 Iterative feedback and reordering

As illustrated in Figure 4.1, our approach starts with a presumption about the operational environment and begins its CWS operation normally in full sampling mode. At first no reordering is performed, i.e., $\pi = (1, 2, 3, \ldots, n)$. After reconstructing the signal from the measurement vector, we apply SOPerm to improve signal sparsity under the newly calculated reordering π^*. This new permutation may lead to a sparser signal vector in the Ψ-domain. The parameter η (see Figure 4.1) determines how many unit vectors from the Ψ-basis are compared by the SOPerm algorithm.

As discussed in Section 4.2.2, a suboptimal permutation can be often found among very first unit vectors of the Ψ-domain because the natural

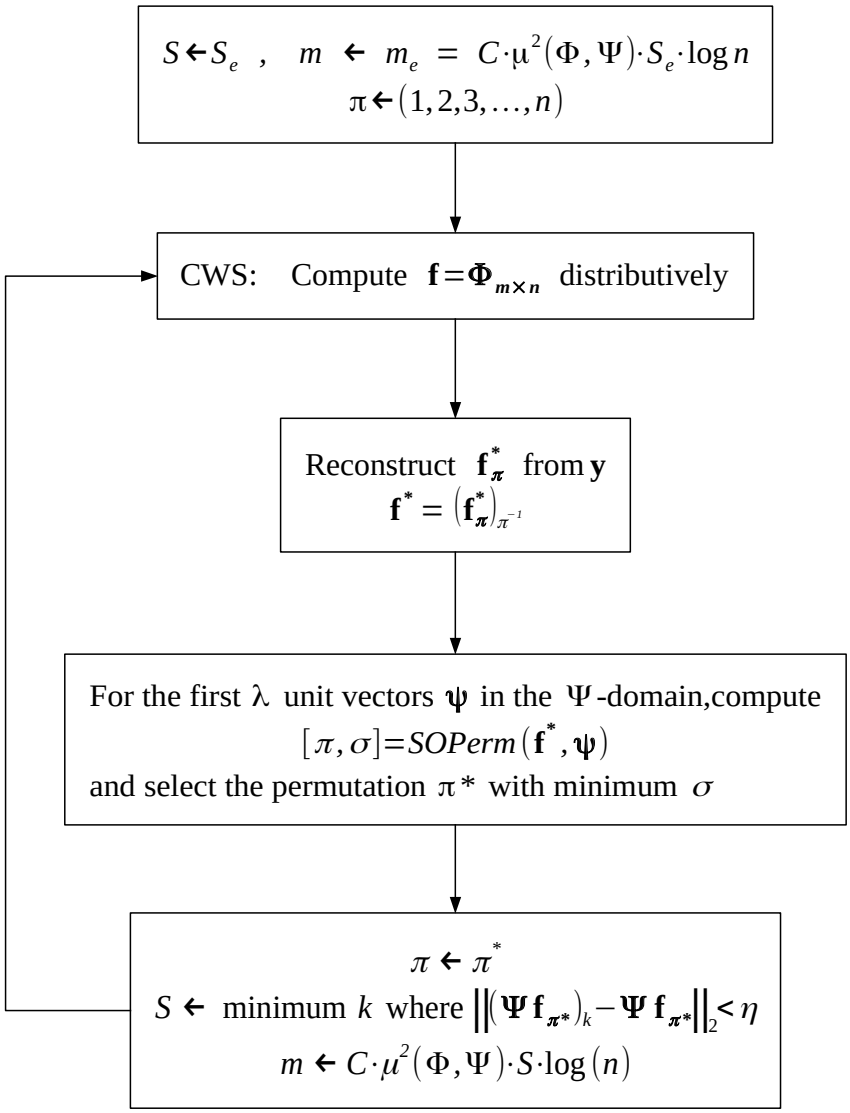


Figure 4.1: Iterative CWS and sample reordering

phenomenon being sampled is expected to be projected to the lower frequency coefficients in its Ψ-transform. The lower the parameter η, the faster our approach computes sub-optimal reordering. The higher the parameter η, the higher is the chance to find a better permutation. However, in search for matching to unit basis vectors among high frequencies, it is less probable to find a better permutation than the one we have found among the lower frequencies.

Parameter $\sigma > 0$ is a threshold which controls that with how much precision we consider a signal to be sparse in Ψ-domain. The smaller the threshold σ, the more accurate is our CWS reconstruction, but at the cost of higher sampling rate. In each cycle, the current reconstructed state of the environment is displayed to the user by mapping back the elements of f_{π^*} under the inverse permutation indexing vector π^{-1}:

$$\mathbf{f}^* = (\mathbf{f}^*_{\pi})_{\pi^{-1}} \tag{4.5}$$

4.5 The impact of sample reordering

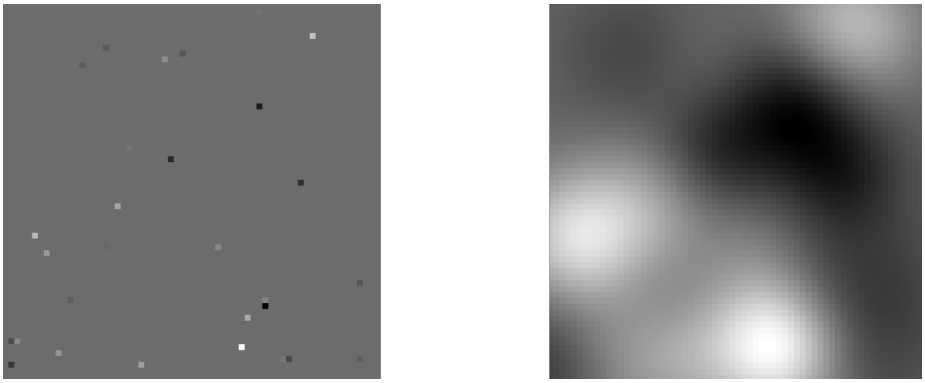
We perform simulations to evaluate our adaptive CS for WSN. After detailing the settings, we present the evaluation results of our reordering algorithms.

4.5.1 Simulation environment

Our simulations are done in MATLAB. The simulation environment is defined as a rectangular area where a set of SNs are randomly distributed. In this simulation, we evaluate the impact of reordering on sparsity of the signal and does not consider every details of the communication protocol. Our prototype scripts are in fact numerical experiments that show how the sparsity of the spatial signal is enhanced under reordering found by SOPerm.

We simulate the behavior of a natural environment and work with the generated synthetic data. As a proof of concept, we consider the air temperature of individual points on a rectangular area. We can view the temperature map of the environment as a 2-D gray-scale picture. To construct such a 2-D image of the temperature map that behaves much like a real temperature map we need to construct a picture whose pixel intensities do not vary sharply along x or y axis.

The following steps show how we construct such a picture with smooth variations of pixel intensity. At the first step, we paint all the pixels with intensity 0. The second step is to set some points of the image to random values other than zero. In our implementation of the environment simulator,



(a) Random points on the temperature map

(b) Gaussian filter applied on (a)

Figure 4.2: Generating synthetic distributed signal

we choose some points on the image, and set them a random real number chosen from a Gaussian random distribution with zero mean and variance 1.0. Finally, we apply a 2-D Gaussian filter on the resulting image.

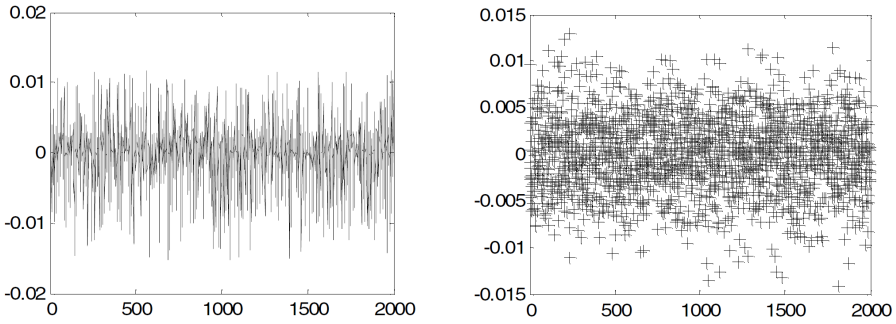
Figure 4.2 depicts our method for a 64×64 image. Figure 4.2(a) shows the random guiding points. Figure 4.2(b) shows the final simulated operational environment after applying a Gaussian filter on Figure 4.2(a).

4.5.2 Impact of reordering on signal compressibility

SNs are distributed randomly on the surface of a temperature map like Figure 4.2(b). Signal vector $\mathbf{f}$ is composed of the intensity of the pixels located at the points where SNs are situated. Our SOPerm algorithm gives us a reordering of $\mathbf{f}$ such that its DCT transform is sparser. Note that in our numerical experiments we have chosen DCT as our Ψ-domain for simplicity, and to avoid dealing with imaginary parts of complex numbers.

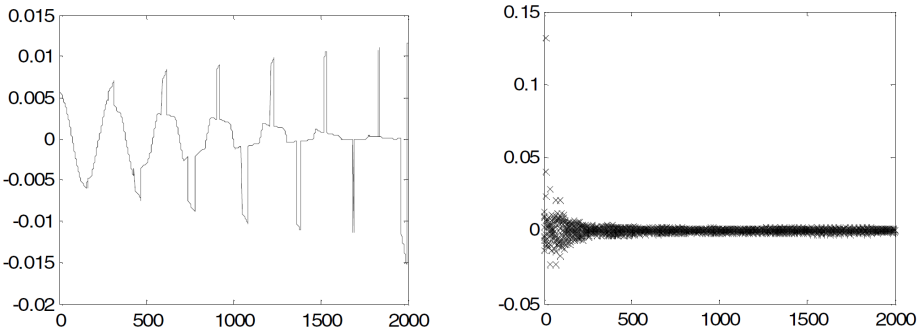
Figure 4.3(a) shows the original discrete signal $\mathbf{f}$ of size 2000 composed of the sensed temperature values according to permutation of samples ordered by the original index (id) of the SNs. Figure 4.3(b) depicts the DCT transform of the actual vector $\mathbf{f}$. Then, we apply SOPerm on $\mathbf{f}$ with regard to the first 20 unit basis vectors in the DCT system, and finally we choose the best permutation among the 20 permutations found by the SOPerm. Let us call the sub-optimal permutation as π^*.

Figure 4.4(a) shows the reordered vector $\mathbf{f}_{\pi^*}$ computed by our SOPerm algorithm and Figure 4.4(b) depicts its DCT transform. It is apparent that $\mathbf{f}_{\pi^*}$ is much sparser under the DCT domain than the original signal $\mathbf{f}$. Note



(a) Indexed by SN id's (b) DCT transform

Figure 4.3: Amplitude of original synthetic signal and its DCT coefficients



(a) Sub-optimally permuted (b) DCT transform

Figure 4.4: Synthetic signal after reordering

that because the signal is synthetically generated and is not based on real data of a physical phenomenon, the amplitude of the signal in the figures are without unit.

4.5.3 Reordering of dynamic signals

Now, we show that the permutation computed for a discrete spatial signal at a specific moment, can be still good enough for upcoming moments in future, given the precondition that the state of the environment does not change very quickly over time. To simulate such a dynamic environment, we have upgraded our synthetic environment described in Section 4.5.1 to simulate changes over time. By randomly moving the points (Figure 4.2(a)) over the image to different directions, we simulate the environment changes. Simultaneously, we pass the image through a Gaussian filter to keep realistic-appearing distribution of temperature that changes over time. Figure 4.5 shows the outcome of such a synthesized animated image.

In Figure 4.5, the points move around the rectangular image randomly in each direction to maximum 10 pixels away. This small steps causes that the environment does not change too quickly over time. In Figure 4.5, the environment changes in the sequence specified by the directed arrows. We have run the simulation for 64 time periods and the images shown in Figure 4.5 are actually 8 snapshots taken every 8 time units. During all this time, 200 SNs are compressively reporting the value of the physical parameter under the points they are located. Therefore, at each time instance, we have a different spatial signal vector. For each vector, we run SOPerm with regard to DCT domain and find a sub-optimal permutation. We keep a history of previously computed sub-optimal permutations. Figure 4.6 shows how old permutations may still lead to sparser DCT transform of the signal at the current time. Equivalently, this means that a suboptimal permutation at a specific time is still sub-optimal for the following time instances.

Figure 4.6 shows that the sparsity of the signal with its normal order (denoted by $\square$) stands always lower than suboptimal permuted signal vector and the signal vector reordered according to previously calculated sub-optimal permutations. At each time instance t, SOPerm is applied on the signal vector $\mathbf{f}_t$ and resulted π_t as a permutation for the signal at time instance t. Sparsity of $(\mathbf{f}_t)_{\pi_t}$ is depicted with (*) on Figure 4.6.

The interesting result is that previously computed suboptimal permutations still make the signal sparser even for signal vectors at times later than the instance they have been computed. However, as it can be seen on the diagram, outdated permutations (previously computed π_t where $t < 30$) do not provide a good reordering anymore. As we approach closer to the present

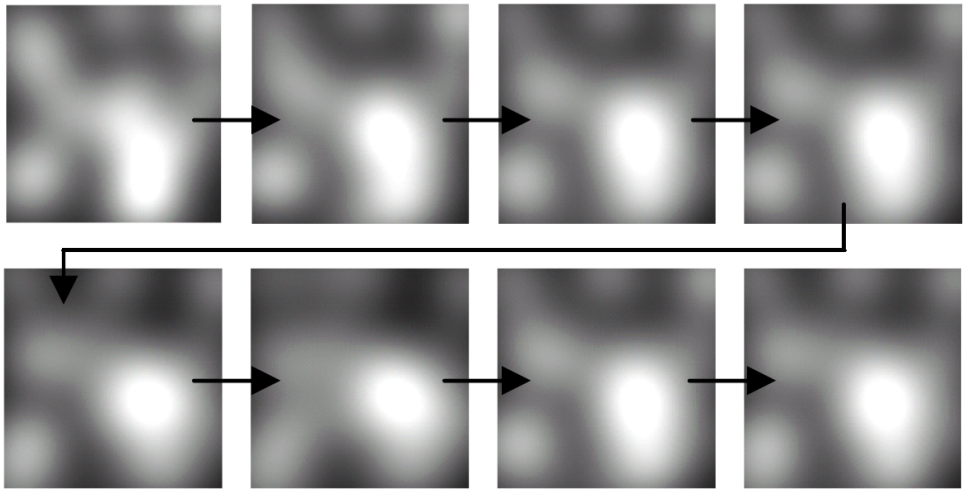


Figure 4.5: Simulating a dynamic synthesized signal varying over time

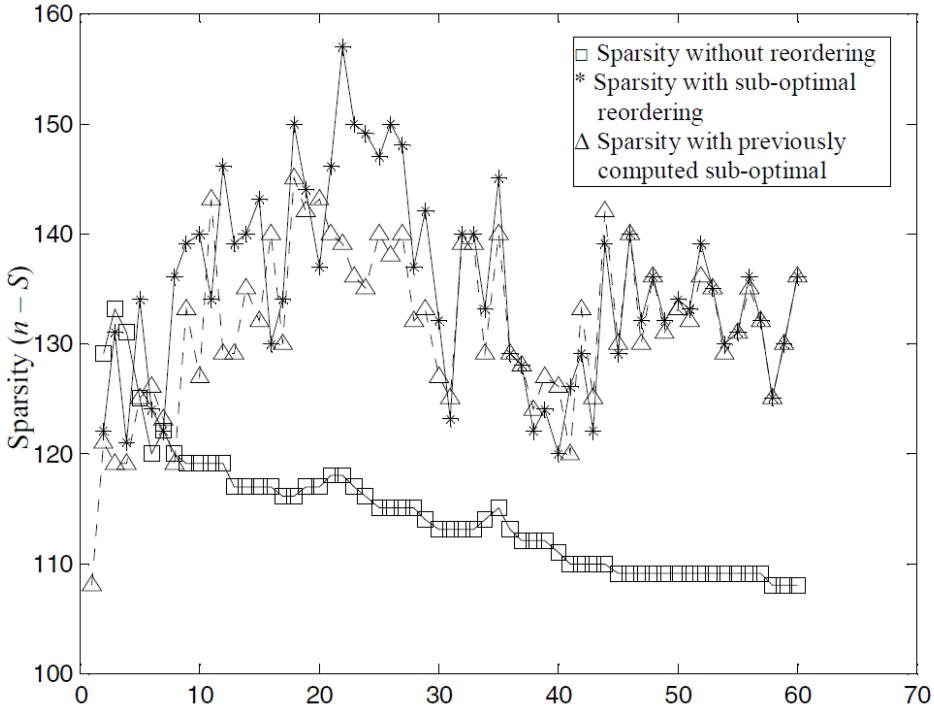


Figure 4.6: Sparsity variations over time

time ($t > 30$ till $t = 64$), we see that previously computed π_t's work as good as the SOPerm permutation which is especially calculated for $\mathbf{f}_{64}$.

4.5.4 Impact of reordering on different WSNs

So far we have run our simulations only for a fixed WSN. Now we vary the number of nodes for $200 \leq n \leq 800$. We first construct synthetic simulated operational environment. For each n, we randomly distribute n SNs over the 2D image representing our environment and acquire spatial signal $\mathbf{f}$ and then we apply SOPerm on $\mathbf{f}$ to see how much improvement in sparsity we can gain. For each of these WSNs, we run the same simulation 100 times and compare the averaged sparsity of discrete spatial signal $\mathbf{f}$ with and without reordering by SOPerm. Figure 4.7 depicts the result of this experiment. The experiment is run with the Fourier domain as our Ψ-domain. This experiment also illustrates the validity of SOPerm algorithm for transformation matrices with imaginary elements. Because Fourier transform matrix contains both real and imaginary values, we can expect a decrease in sampling rate equal to half of the sparsity improvement illustrated in Figure 4.7. During the test, the threshold η is set to 0.01. With the reordering resulted from SOPerm algorithm, environmental spatial signal stands always sparser than the original signal that is not reordered using SOPerm. This overall experiment shows the generality of our proposed model. This method can to be used to enhance CWS and decrease sampling rate and energy consumption while delivering the same quality of environmental signal reconstruction.

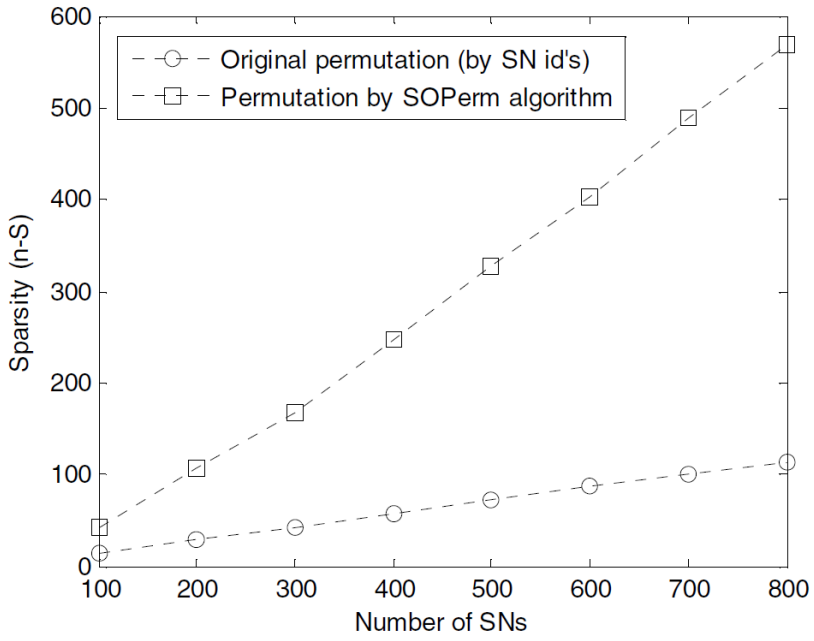


Figure 4.7: Comparing sparsity of the spatial signal with and without reordering

Chapter 5

Spatiotemporal Compressive Sampling

Several studies have shown that the sensor observations are both spatially and temporally correlated Vuran et al. [2004]. Therefore, data acquisition techniques can achieve better efficiency by exploiting redundancy and compressing the raw data by considering spatial and temporal compression at the same time.

Spatial CS for WSNs was proposed in Compressive Wireless Sensing (CWS) Bajwa et al. [2006] and developed to Compressive Data Gathering (CDG) Luo et al. [2010, 2009]. Distributed Compressive Sensing (DCS) Duarte et al. [2006] extends CS-based spatial sampling techniques for WSNs to temporal domain by considering temporal as well as spatial correlations between sensor observations. As detailed in Section 3.3.3, DCS model is best suited for WSNs with star topology.

We address two key shortcomings of the existing CS-based spatiotemporal sampling techniques:

- Implementing DCS on multi-hop WSNs with tree topology leads to unbalanced energy consumption and exhausted nodes.
- Existing spatiotemporal CS techniques for WSN usually consider non-continuous sampling periods. Consequent intervals are treated separately when acquiring measurements and recovering the signal.

We show that temporal correlation of overlapping sampling periods allows for non-interruptive and more efficient measurement and signal recovery. We enhance CS-based spatiotemporal sampling in multi-hop WSNs by:

1. A new spatiotemporal CS model that puts a balanced computation and communication load on all SNs,

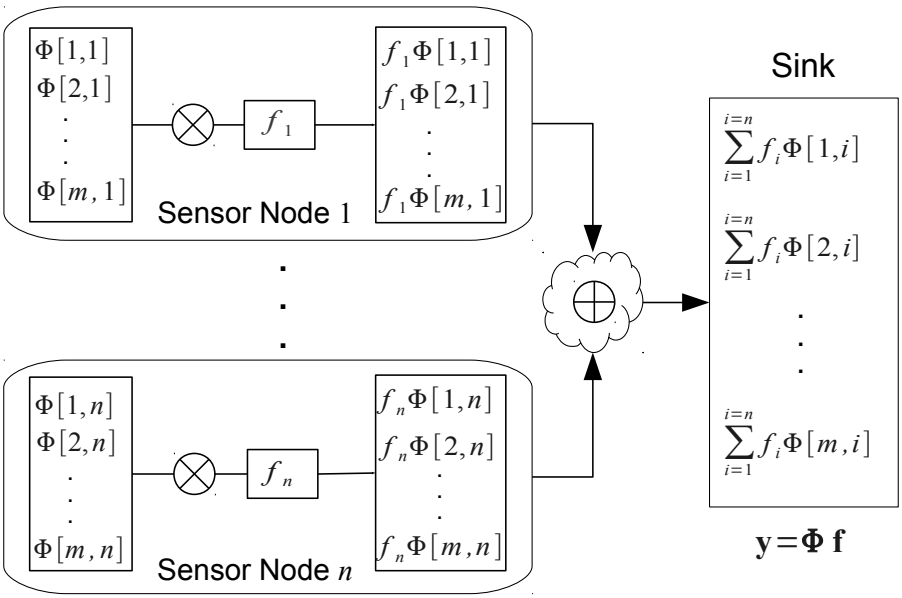


Figure 5.1: Distributed spatial sampling using CWS in star topology

2. Applying the concept of *sampling window* to streamline the process of compressive measurement and signal reconstruction. Our proposed method progressively acquires compressive measurements from the network and reconstructs the signal from a set of consequent spatiotemporal measurements.

Recalling the concepts of Chapter 3, Figure 5.1 shows a simple configuration in which n SNs are directly connected to the sink. The ith SN can access the ith column of the matrix $\mathbf{\Phi}$. This column might be embedded into the SN before deployment or calculated on the fly when using random measurement matrices. Note that since addition is a commutative operation, the summation can be calculated in any order. For example, Figure 5.2 depicts the distributed computation of the measurement vector in a WSN with chain topology. Evidently, any tree topology can be constructed by a combination of star and chain topologies. Therefore, CWS can be applied to any topology like star topology, or multi-hop WSNs pertaining tree or chain topologies. The cloud shape enclosing the summation operator signifies that any tree-based topology can compute the measurement vector $\mathbf{y}$ and deliver it to the sink.

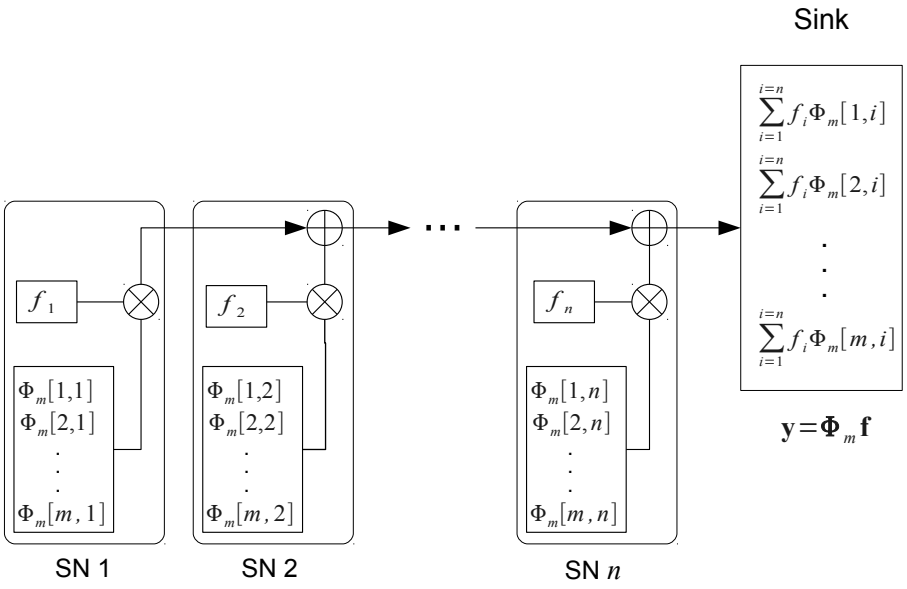


Figure 5.2: Distributed spatial sampling using CWS in chain topology

5.1 Extending CS to temporal domain

If we take another look at Equation (3.2), we realize that CS is also very efficient in number of required measurements. The number of required compressive measurements, namely the parameter m in Equation (3.2), grows logarithmically with the dimension of the signal $\mathbf{f}$. This means that higher performance of the CS-based methods is intrinsically visible for high dimensional signals when N is large enough. In plain spatial CS methods like CWS, the dimension of the spatial signal is equal to the number of SNs. Therefore, simple spatial CS might be less useful for small- to medium-scale WSNs. If we extend our model to the temporal domain, we can exploit the desirable logarithmic cost growth even in small- and medium-scale WSNs by increasing the temporal sampling rate of individual SNs. This is the chief motive for our spatiotemporal method.

5.1.1 Block-diagonal measurement matrix in DCS

The DCS technique models jointly sparse signals in a distributed system and introduces a new algorithm suited for recovering jointly-sparse signals. The most important result of DCS is that, it is possible to recover a jointly sparse

signal from much fewer number of measurements than when applying CS individually on every SN. It exploits spatial as well as temporal correlation between samples to reduce the overall number of measurements. DCS differs from CS not only in recovery algorithm, but also in the measurement matrix. In DCS, the measurement matrix is a block-diagonal matrix composed of several *temporal* measurement sub-matrices. DCS is perfect for WSNs with star topology where every SN can directly send data to the sink. Assume that the ith SN is recording r_i samples every T time units and build up a vector $\mathbf{f}'_i \in \mathbb{R}^{r_i}$. Temporal values of each SN produce such a discrete temporal signal and all of them together form a discrete spatiotemporal signal $\mathbf{f}' = [{\mathbf{f}'_1}^{tr} {\mathbf{f}'_2}^{tr} \cdots {\mathbf{f}'_n}^{tr}]^{tr}$ of size $N = r_1 + r_2 + ... + r_n$ where $[\cdot]^{tr}$ is the transpose operator. The measurement vector $\mathbf{y}' \in \mathbb{R}^m$ is also composed of n subvectors such that:

$$\mathbf{y}' := \begin{pmatrix} \mathbf{y}'_1 \\ \mathbf{y}'_2 \\ \vdots \\ \mathbf{y}'_n \end{pmatrix} = \begin{pmatrix} \mathbf{\Phi}'_1 & & & \\ & \mathbf{\Phi}'_2 & & \\ & & \ddots & \\ & & & \mathbf{\Phi}'_n \end{pmatrix} \begin{pmatrix} \mathbf{f}'_1 \\ \mathbf{f}'_2 \\ \vdots \\ \mathbf{f}'_n \end{pmatrix} \tag{5.1}$$

where each $\mathbf{\Phi}'_i$ $i \in \{1, 2, \cdots, n\}$ has r_i columns. Having

$$\mathbf{\Phi}' := \begin{pmatrix} \mathbf{\Phi}'_1 & & & \\ & \mathbf{\Phi}'_2 & & \\ & & \ddots & \\ & & & \mathbf{\Phi}'_n \end{pmatrix} \quad \text{and} \quad \mathbf{f}' = \begin{pmatrix} \mathbf{f}'_1 \\ \mathbf{f}'_2 \\ \vdots \\ \mathbf{f}'_n \end{pmatrix}, \tag{5.2}$$

The measurement operation can be formulated by $\mathbf{y}' = \mathbf{\Phi}'\mathbf{f}'$.

Unbalaned operation of DCS in multi-hop WSNs

Employing DCS in a multi-hop WSN, leads to unbalanced communication overhead that eventually causes network partitioning and or coverage drops due to the depletion of the batteries of more active nodes. Figure 5.3 shows what happens when transmitting vector $\mathbf{y}'$ in Equation (7.4) over a WSN with chain topology. Every SN calculates its own component of the measurement vector $\mathbf{y}'$. In a multi-hop topology each component is treated separately as a data packet. The message length increases as the data packets approach the sink. New sub-vector components must be attached to the messages received from previous hops. Apparently, DCS does not lead to a balanced data acquisition mechanism for multi-hop WSNs.

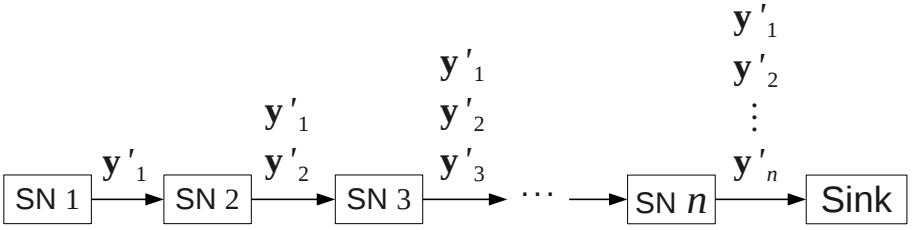


Figure 5.3: Measurement mechanism of DCS in multi-hop WSNs

5.1.2 Balanced spatiotemporal CS for multi-hop WSNs

Yap et al. have shown that block-diagonal random measurement matrices can perform as good as dense random measurement matrices in CS signal acquisition and recovery Yap et al. [2011]. Similar to DCS, we also use a block-diagonal measurement matrix. However, we propose a different model for spatiotemporal signals and a new structure of the measurement matrix. Let $\mathbf{\Phi}_t$ and $\mathbf{f}_t \in \mathbb{R}^n$ denote the measurement matrix and the spatial signal at time t, respectively. The measurement vector at time t will be $\mathbf{y}_t = \mathbf{\Phi}_t \mathbf{f}_t$. We perform the measurement process T times and then recover the spatiotemporal signal $[\mathbf{f}_1{}^{tr}\, \mathbf{f}_2{}^{tr}\, \cdots\, \mathbf{f}_T{}^{tr}]^{tr}$ from υ such that:

$$\upsilon := \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_T \end{pmatrix} = \begin{pmatrix} \mathbf{\Phi}_1 & & & \\ & \mathbf{\Phi}_2 & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_T \end{pmatrix} \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_T \end{pmatrix} \tag{5.3}$$

where for every $1 \leq t \leq T$, $\mathbf{\Phi}_t$ has m_t rows and n columns.

There is a fundamental difference between measurement matrices in Equation (5.2) and Equation (5.3). In Equation (5.2), each block $\mathbf{\Phi}'_i$ $i \in \{1, 2, \cdots, n\}$ corresponds to a single SN sampling over a period T. In Equation (5.3), each block $\mathbf{\Phi}_t$ $t \in \{1, 2, \cdots, T\}$ corresponds to spatial samples acquired by all n SNs at time instance t. Note that in contrast to DCS, each SN in our model transmits $m = m_1 + m_2 + \cdots + m_T$ measurements to deliver the measurement vector υ to the sink. Therefore, our model still benefits from the balanced energy consumption like CWS. Remember from Figure 5.3 that in the extreme case of chain topology, the number of transmissions in DCS increases in order of $O(n^2)$ as the measurements traverse the network hop by hop to approach the sink.

We define the size of a WSN to be the number of its SNs and is denoted by n. Nodes that do not record or sense the physical parameter of interest, are

not counted [1]. As mentioned before, we denote by T the sampling interval of a WSN , that is the period in which all the compressive spatiotemporal measurements must be acquired from SNs. The spatiotemporal signal recovery is done at the end of every sampling interval. Sampling rate of the ith SN is denoted by r_i and is equal to the number of samples to be recorded by the ith SN in sampling interval T. Sampling rates of the SNs are not necessarily equal and may differ from one SN to another one according to the temporal accuracy required at the position where the SN is located.

Definition 7. *Partial spatiotemporal signal recorded by the i^{th} SN in every sampling interval is a discrete signal vector $\mathbf{g}_i$ composed of the values recorded in a single sampling interval by the i^{th} SN. Therefore, the j^{th} entry of vector $\mathbf{g}_i$ in a specific sampling interval is the value recorded by the i^{th} SN at time instance $(j/r_i)T$ just after the beginning of that sampling interval.*

The dimension of the spatiotemporal signal vector $\mathbf{g} \in \mathbb{R}^N$ is the number of its entries, namely N, which is equal to the sum of the partial samples by every SNs, i.e, $N = \sum_{i=1}^{n} r_i$. Total spatiotemporal signal or simply the spatiotemporal signal vector $\mathbf{g} \in \mathbb{R}^N$ is made by joining the partial spatiotemporal signals (which are basically temporal signals) recorded by all SNs. More formally, $\mathbf{g} = [\mathbf{g}_1^T \mathbf{g}_2^T \mathbf{g}_3^T \dots \mathbf{g}_n^T]^T$.

Definition 8. *Partial measurement matrix $\Phi_m^{(i)}$ for the ith SN is an $m \times r_i$ matrix consisting of the kth, $k+1$st, $k+2$nd, $\dots, (k+r_i-1)$th columns of the measurement matrix Φ_m where $k = 1$ when $i = 1$ and $k = \sum_{j=1}^{i-1} r_j$ otherwise.*

Definition 9. *Partial measurement vector $\mathbf{y}_i \in \mathbb{C}^m$ by the i^{th} SN is equal to $\Phi_m^{(i)} \mathbf{g}_i$ where $\Phi_m^{(i)}$ and $\mathbf{g}_i$ are the partial measurement matrix for the i^{th} SN and partial spatiotemporal signal recorded by the i^{th} SN according to Definitions 8 and 7 respectively.*

Note that all the partial measurement vectors have the same dimension and when added together, result in the linear transform $\Phi_m \mathbf{g}$. Formally speaking, $\sum_{i=1}^{n} \mathbf{y}_i = \sum_{i=1}^{n} \Phi_m^{(i)} \mathbf{g}_i = \mathbf{y} = \Phi_m \mathbf{g}$. Our spatiotemporal model still benefits from balanced energy consumption in CWS methodology. Computation and memory complexity required by the ith SN is proportional to its sampling rate r_i. Similar to typical spatial CWS, if random measurement matrices are applied here, memory complexity reduces to a constant when using a common pseudo-random number generator with a predetermined seed known to sink.

[1] All nodes of a WSN are not necessarily sensor nodes, some of them may function as message forwarding or administrative nodes.

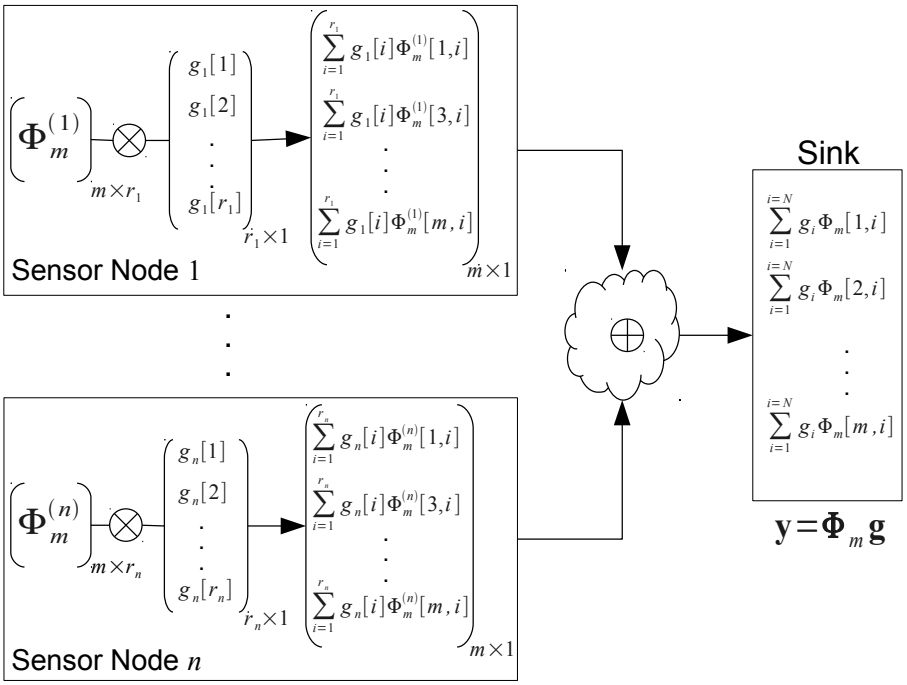


Figure 5.4: Spatiotemporal sampling model for multi-hop WSN

Figure 5.4 illustrates our spatiotemporal CS model. Comparing to Figure 5.1 and Figure 5.2, we see that here the size of the WSN, namely n, is not necessarily equal to the dimension of the spatiotemporal signal, namely N. In fact, typical spatial CWS is a special case of our spatiotemporal CWS with $r_1 = r_2 = r_3 = \cdots = r_n = 1$. Every SN accumulates the values sampled by the node itself and the received samples from children on the buffer location pointed by the corresponding index of that sample. Thus the nodes do not need to be exactly synchronized. When a measurement from lower levels of the tree arrives with delay, it is still possible to search in buffer for its corresponding index and accumulate its value with the correct location. A buffer counter or index variable takes care of the number of accumulated values. When it reaches the total number of samples per node plus the number of its children, that buffer value is sent to the next hop. That buffer entry will be then marked as free for future use. This process continues until the aggregation is completed throughout the network.

Evaluations show that our model for the spatiotemporal signal leads to a more compressible representation of the signal especially when the spatiotemporal signal is acquired over longer periods. As mentioned earlier, the logarithmic growth of sampling cost in CS encourages us to try acquiring

more samples over longer periods. This way, we can acquire a spatiotemporal signal with higher dimension which entails even better compressibility.

First, we formally define the efficiency of a CS-based signal acquisition method in WSNs. This efficiency is directly related to the compressibility of the signal. We will assess our model of spatiotemporal signals by investigating the level of compressibility that we achieve by extending CS to the temporal domain over longer sampling periods. We also need to exactly define *sampling period* and *sampling round*.

Definition 10. *Sampling period of length T is composed of T sampling rounds. Each sampling round occurs at a discrete time instance on a regular basis. During a sampling round, all SNs record the sensed value at that time instance. Compressive measurements are then calculated distributively from the recorded values and transmitted over the multi-hop WSN to the sink.*

Definition 11. *The efficiency of the signal acquisition in a CS-based system is denoted by η defined as $\eta = (N - m)/N$ where m is the number of measurements according to Equation* (3.2) *and N is the dimension of the spatiotemporal signal.*

When more measurements need to be acquired for a fixed N, the efficiency will be lower. If the signal is recoverable from fewer measurements, the efficiency increases. According to Equation (3.2), for an efficiency of η, we require S/N to be less than $(1-\eta)/(C[\mu^2(\Phi, \Psi)] \log N)$. We have investigated how this efficiency can be achieved by running numerical experiments on real-world data sets. Evaluations are done using real-world data of air temperature values collected by the LUCE (Lausanne Urban Canopy Experiment) WSN deployment at EPFL LUC [2008]. The dashed curve in Figure 5.5 shows S/N when efficiency is roughly equal to 90% and recovery error is bounded to $\pm 1°C$ per SN. We have calculated S/N under Haar wavelet and DCT transformation for spatiotemporal signal with different sampling periods. We see that when the sampling period T increases, the S/N drops quickly under the limit that satisfies $\eta = 90\%$. Next, we will see how higher compressibility (i.e., lower S/N) leads to higher efficiency. The trade-off here is the more delay required to sample signals over longer periods. We see that for a slight increase in T, the efficiency can be significantly improved. Therefore, we expect not so much delay in a real world implementation.

5.2 The concept of sampling window

So far, we have seen improvements in compressibility of spatiotemporal signals when using block-diagonal random measurement matrices over longer

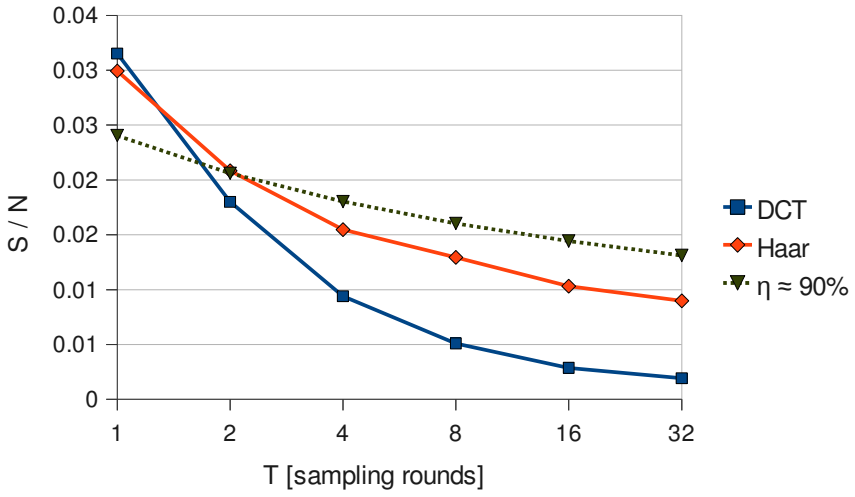


Figure 5.5: Effect of longer sampling periods on spatiotemporal compressibility

sampling periods. The trade-off is a delay equal to T sampling rounds. Increasing T usually leads to better compressibility of the spatiotemporal signal and improves efficiency. However, the signal reconstruction is delayed by T sampling rounds. Here, we introduce the concept of *sampling window* and show that the delay affects measurements acquisition and signal recovery only at initialization. When T periods are over, only in-network communication delays affect recovery of every consequent $\mathbf{f}_u$ where $u > T$. Assume for any time instance $\tau \geq T$, the measurement vectors $\mathbf{y}_{\tau-T+1}$, $\mathbf{y}_{\tau-T+2}$, $\cdots$, $\mathbf{y}_\tau$ are delivered to the sink. Therefore, all of the vectors $\mathbf{f}_{\tau-T+1}$, $\mathbf{f}_{\tau-T+2}$, $\cdots$, $\mathbf{f}_\tau$ which describe the sate of the environment during the interval $[\tau - T + 1, \tau]$ are recoverable from υ_τ where:

$$\upsilon_\tau := \begin{pmatrix} \mathbf{y}_{\tau-T+1} \\ \mathbf{y}_{\tau-T+2} \\ \vdots \\ \mathbf{y}_\tau \end{pmatrix} = \begin{pmatrix} \mathbf{\Phi}_{\tau-T+1} & & & \\ & \mathbf{\Phi}_{\tau-T+2} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_\tau \end{pmatrix} \begin{pmatrix} \mathbf{f}_{\tau-T+1} \\ \mathbf{f}_{\tau-T+2} \\ \vdots \\ \mathbf{f}_\tau \end{pmatrix}. \quad (5.4)$$

In the next sampling round, $\mathbf{f}_{\tau+1}$ is sensed by the SNs and should be reconstructed at the sink. Of course we do not want to perform typical CWS from scratch at sampling round $\tau+1$. Having previous measurements using block-diagonal measurement matrix in Equation (3.5), we can recover the signal at the next sampling round from fewer measurements compared with typical spatial CWS. Our spatiotemporal measurement method should calculate the new measurement vector:

$$\upsilon_{\tau+1} := \begin{pmatrix} \mathbf{y}_{\tau-T+2} \\ \mathbf{y}_{\tau-T+3} \\ \vdots \\ \mathbf{y}_{\tau+1} \end{pmatrix} = \begin{pmatrix} \mathbf{\Phi}_{\tau-T+2} & & & \\ & \mathbf{\Phi}_{\tau-T+3} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_{\tau+1} \end{pmatrix} \begin{pmatrix} \mathbf{f}_{\tau-T+2} \\ \mathbf{f}_{\tau-T+3} \\ \vdots \\ \mathbf{f}_{\tau+1} \end{pmatrix}. \quad (5.5)$$

Comparing Equations (5.4) and (5.5), we observe that if υ_τ is already present at the sink, then we only need $\mathbf{y}_{\tau+1}$ to be delivered to the sink in order to make $\upsilon_{\tau+1}$. In fact, a *sampling window* of T allows us to efficiently recover the spatiotemporal signal at any sampling round. Note that in this case, to have $\mathbf{y}_{\tau+1}$ (and consequently $\upsilon_{\tau+1}$) at the sink, only $m_{\tau+1}$ more measurements are needed which would be lower than when running CWS on sampling round $\tau+1$. Moreover, all SNs transmit an equal amount of data proportional to $m_{\tau+1}$.

5.2.1 Benefits of sampling window

Here, we clarify the advantage of our sampling window mechanism compared with the state of the art CS-based spatiotemporal signal acquisition techniques. Sampling window as described above allows for recovering the spatiotemporal signal from a history of the current and previously acquired measurements. We recall that CWS only considers instantaneous spatial measurements and DCS operates over disjunct intervals. The advantages of our proposed sampling window are twofold: First, total number of measurements, namely $\sum_{i=1}^{i=T} m_{\tau-T+i}$ is much less than when running CWS separately for T sampling rounds because considering temporal as well as spatial correlations leads to better compressibility and hence more efficient signal acquisition. Second, acquiring measurements and recovering the signal are done seamlessly with a much less delay. Remember that DCS recovers spatiotemporal signals after a delay proportional to T when all measurements

of the last T sampling rounds are received at the sink. In particular, for an extensive multi-hop WSN, our model decreases the delay by a factor of T. Note that here, *delay* refers to the time required to acquire measurements from the network and not the time required by the recovery algorithm to reconstruct the spatiotemporal signal. The sink is assumed to have enough computation capabilities to recover the signal. Many efficient CS reconstruction algorithms such as orthogonal matching pursuit are developed to recover the signal in a timely manner Tropp and Gilbert [2007].

This result has an important application in our work to decrease the delay during signal acquisition. Assume we use a block-diagonal Gaussian random measurement matrix $\mathbf{\Phi}_m$ with m rows and $N = nT$ columns, such that each block corresponds to a sampling period, i.e.:

$$\mathbf{\Phi}_m = \begin{pmatrix} \mathbf{\Phi}_{m_1} & & & \\ & \mathbf{\Phi}_{m_2} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_{m_T} \end{pmatrix} \tag{5.6}$$

where $m_1 + m_2 + \cdots + m_T = m$ and for each $i \in 1, ..., T$, Φ_{m_i} has n columns. This operation is basically equivalent to applying spatial measurement matrices with m_1, m_2, ... , m_T rows over T sampling periods. The advantage of using such a measurement matrix is that measurements can be progressively transmitted to the sink and there is no need to wait for the whole sampling period to finish and then send the measurements.

5.2.2 Detecting events in sampling window

Luo et al. have proposed an effective method to detect and handle sparse abnormal sensor readings using overcomplete dictionaries Luo et al. [2010, 2009]. We apply a similar method for detecting abnormal events in order to trigger notification about potentially harmful situations. What we are especially interested in, is how our sliding sampling window can detect the new events occurring in the most recent sampling round. Assume few SNs record unexpected values at time u where $u \geq T$. We can decompose $\mathbf{f}_u$ into two vectors $\mathbf{f}_c$ and $\mathbf{f}_e$ such that $\mathbf{f}_u = \mathbf{f}_c + \mathbf{f}_e$ where $\mathbf{f}_e$ is the sparse abnormal innovation vector. We know that $[\mathbf{f}^{tr}_{u-T+1} \quad \mathbf{f}^{tr}_{u-T+2} \quad \cdots \quad \mathbf{f}^{tr}_{u-1} \quad \mathbf{f}^{tr}_{c}]^{tr} = \mathbf{\Psi}\xi$ is compressible in the Ψ-domain. Substituting $\mathbf{f}_u$ with $\mathbf{f}_c + \mathbf{f}_e$ we will have:

$$\mathbf{g} := \begin{pmatrix} \mathbf{f}_{u-T+1} \\ \mathbf{f}_{u-T+2} \\ \vdots \\ \mathbf{f}_u \end{pmatrix} = \begin{pmatrix} \mathbf{f}_{u-T+1} \\ \mathbf{f}_{u-T+2} \\ \vdots \\ \mathbf{f}_c + \mathbf{f}_e \end{pmatrix} = \begin{pmatrix} \mathbf{\Psi} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \xi \\ \epsilon \end{pmatrix} \tag{5.7}$$

such that $\epsilon = [\mathbf{0}^{tr} \quad \mathbf{f}_e{}^{tr}]^{tr}$, where $\mathbf{0}$ means a zero vector of size $nT - n$. For recovering the original signal as well as detecting abnormal events, we solve a convex optimization problem similar to that in Theorem 1. From Equation (5.7), we see that $[\xi^{tr} \ \epsilon^{tr}]^{tr}$ is compressible under overcomplete system $\mathbf{\Psi}' = [\mathbf{\Psi I}]$. Assume:

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Phi}_{u-T+1} & & & \\ & \mathbf{\Phi}_{u-T+2} & & \\ & & \ddots & \\ & & & \mathbf{\Phi}_u \end{pmatrix} \tag{5.8}$$

and $\mathbf{z} = \mathbf{\Lambda g}$ is the spatiotemporal measurement vector. If we solve the l_1 minimization problem:

$$\hat{\mathbf{x}}' = \underset{\tilde{\mathbf{x}}' \in \mathbb{R}^{2N}}{\operatorname{argmin}} \|\tilde{\mathbf{x}}'\|_1 \quad \text{subject to} \quad \mathbf{z} = \mathbf{\Lambda \Psi}' \tilde{\mathbf{x}}', \tag{5.9}$$

then it is possible to recover the original spatiotemporal signal $\mathbf{g}$ (including the abnormal samples). The recovered signal of $\mathbf{g}$ will be $\hat{\mathbf{g}} = \mathbf{\Psi}' \hat{\mathbf{x}}'$. Here, $\hat{\mathbf{x}}' \in \mathbb{R}^{2N}$ and $\mathbf{\Psi}'$ has $2N$ columns.

Note that when using a progressive sampling window, we are mainly interested in detecting events at the most recent sampling round. In this case, we see that the abnormal sensor readings are not uniformly distributed over the spatiotemporal signal. The abnormal observations only affect the last chunk of the spatiotemporal signal vector.

5.2.3 Evaluation of the sampling window technique

We have evaluated our proposed methods using real-world data collected by the LUCE WSN deployment at EPFL LUC [2008]. The ambient temperature values of 64 SNs are used as the physical parameter for evaluating our model. In the LUCE dataset, some records were missing or too desynchronized, i.e. the sampling rounds of the SNs were not aligned. Therefore, we have pre-processed the dataset while preserving the attributes of the spatiotemporal signal to have a synchronized data set which is suitable for testing our model. We assume that measurements are calculated while transferring to the sink using a reliable hop-by-hop transport protocol Stann and Heidemann [2003].

Figure 5.6 shows the accuracy of signal recovery using a block-diagonal measurement matrix as described in Equation (5.6). Figure 5.6(a) and Figure 5.6(b) illustrate the results using DCT and Haar wavelet as the the compressive basis respectively. This means that the Ψ-domain in Figure 5.6(a) and Figure 5.6(b) is DCT and Haar respectively. For each compressive domain,

we have tested the measurement and recovery for different sampling window lengths. Parameter T in Equation (5.6) represents the initialization delay as well as the width of the sampling window of our spatiotemporal sampling model for multi-hop WSNs. The X-axis represents the ratio of the number of measurements to the number of all spatiotemporal samples, namely m/N where m and N are the same parameters as used in Equation (3.2). The accuracy of the signal reconstruction is measured by the Signal to Noise Ratio (SNR). The Y-axis represents the SNR in decibels (dB).

We observe that in all cases, the quality of signal reconstruction generally improves as the ratio m/N increases. For smaller values of m/N the system can achieve better efficiency as defined in Definition 11.

The evaluations are first done for $T = 1$ which is basically equivalent to the plain spatial sampling case. Then, the width of the sampling window is increased and the evaluation is repeated for $T = 2$, $T = 4$ and $T = 8$. For larger T, we see that higher signal reconstruction accuracy is possible for lower m/N. This means that a higher reconstruction accuracy can be achieved more efficiently if the width of the sampling window T increases.

We have selected these SNs at random and increased their recorded value at the last sampling round to above 100 degrees Celsius. This may resemble a fire starting in the environment that is sensed by three SNs. The sampling window during this simulation was set to $T = 8$.

Figure 5.7 illustrates how the compressive presentation of the spatiotemporal signal is distorted in presence of abnormal sensor readings. The spatiotemporal signal is measured using a block-diagonal random measurement matrix as described in Equation (5.6). Figure 5.7(a) and Figure 5.7(b) show the projection of the signal contaminated with abnormal readings on DCT and Haar wavelet respectively. The DCT projection is not compressible any more. Haar wavelet projection leads to a more compressible representation, since it can preserve hard edges in the signal better than DCT. Now, we use an overcomplete system $\mathbf{\Psi}' = [\mathbf{\Psi I}]$ to recover the compressible projection. We use two systems $\mathbf{\Psi}'_{\text{DCT}} = [\mathbf{\Psi}_{\text{DCT}}\mathbf{I}]$ and $\mathbf{\Psi}'_{\text{Haar}} = [\mathbf{\Psi}_{\text{Haar}}\mathbf{I}]$ where $\mathbf{\Psi}_{\text{DCT}}$ and $\mathbf{\Psi}_{\text{Haar}}$ represent DCT and Haar wavelet bases respectively. Figure 5.7(c) and Figure 5.7(d) show the recovered compressive projections on $\mathbf{\Psi}'_{\text{DCT}}$ and $\mathbf{\Psi}'_{\text{Haar}}$ respectively. Naturally, we observe that the projections of the spatiotemporal signal on the overcomplete bases are much more compressible that typical DCT and Haar bases.

The SNs in LUCE deployment are not synchronized. Therefore, we have averaged over longer periods to achieve a virtual dataset from a synchronized network. The averaged value for each SN over a short period of time, represents the value of a single time spot in the dataset that we have used as the input to our simulating software. Data of ambient temperature sensor is fed

to the simulator as the testbed input.

Higher compressibility means that less measurements are required to recover the signal with comparable accuracy, or in other words, it is possible to achieve better signal reconstruction quality from the same number of random measurements. With low level of compressibility that typical DCT and Haar projections provide, it is impossible to recover the signal and detect abnormal sensor readings. Using overcomplete dictionaries, the signal could be recovered and the abnormal events are localized. Figure 5.8(a) and Figure 5.8(b) show the recovered signal at the last (8th) sampling round using $\mathbf{\Psi}'_{\text{DCT}}$ and $\mathbf{\Psi}'_{\text{Haar}}$ overcomplete systems respectively. Note that when the sampling window is filled and the measurement process is streamlined, such abnormal events can be detected very shortly as only the last measurement vector is needed to be reported to the sink.

We have also compared our sampling method with that of DCS. Figure 5.9 shows the same results as discussed in explanation of Figure 5.6 for the DCS measurements. The signal reconstruction quality using our proposed block-diagonal spatiotemporal measurement matrix is comparable to signal reconstruction using spatiotemporal model of DCS. More interestingly, we have observed that block-diagonal matrices lead to almost the same performance of dense measurement matrices as expected in Yap et al. [2011]. Figure 5.10 shows the signal reconstruction quality from measurement vectors produced by dense Gaussian random measurement matrices. The SNR is closely comparable to schemes with block-diagonal measurement matrices like DCS and our spatiotemporal block-diagonal model for CS measurements in multi-hop WSNs.

5.3 Chapter summary

In this chapter, we have introduced our spatiotemporal extension to CS-based data collection in WSN. It provides the advantage over DCS that it puts a balanced load on the network regardless of the network topology. The amount of data sent by each SN is proportional to its sampling rate. Furthermore, by applying the concept of signal recovery using overcomplete dictionaries, it is possible to detect the abnormal sensor readings that may be caused by faulty SNs or an unexpected event.

We also introduced the concept of sampling window. A sampling window defines a certain time period in which the samples are being gathered from the network. The sampling window then slides forward to collect more data when time passes. The newly sensed data and previously gathered data are involved in the signal recovery process. This makes the overall scheme

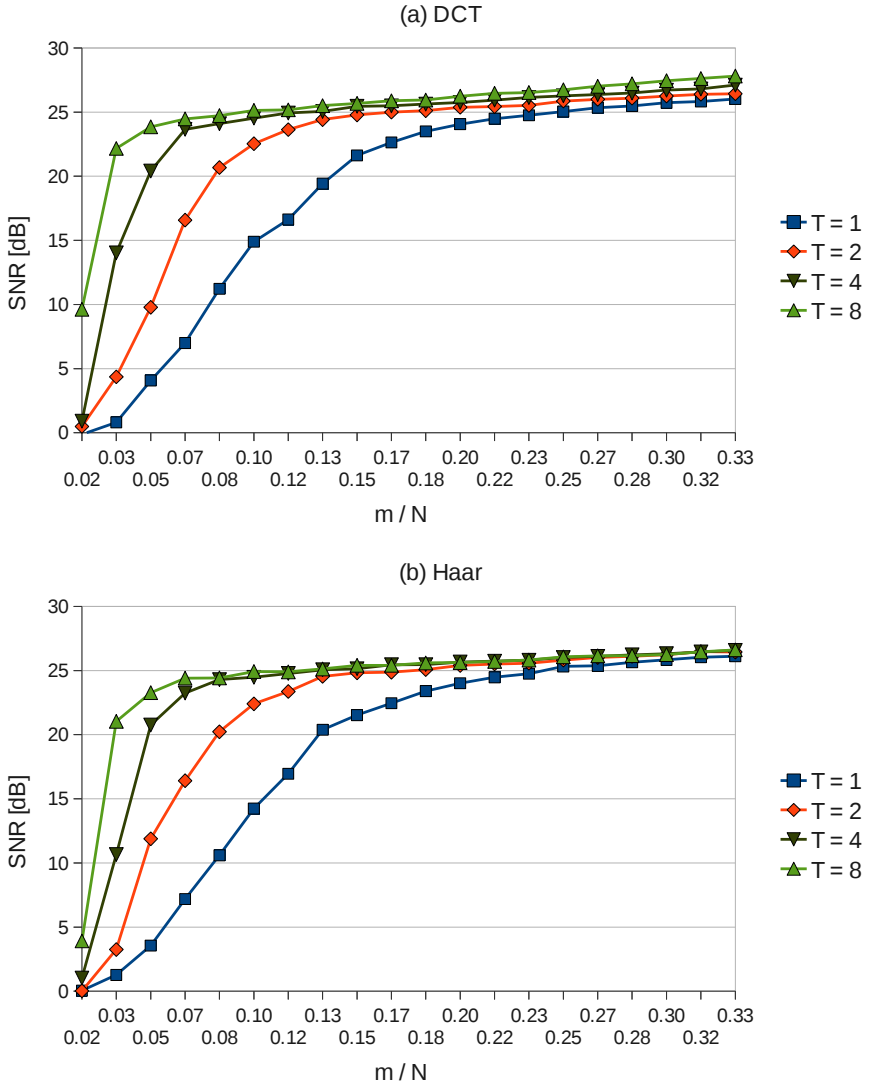


Figure 5.6: Accuracy of multi-hop spatiotemporal CS using block-diagonal measurement matrix

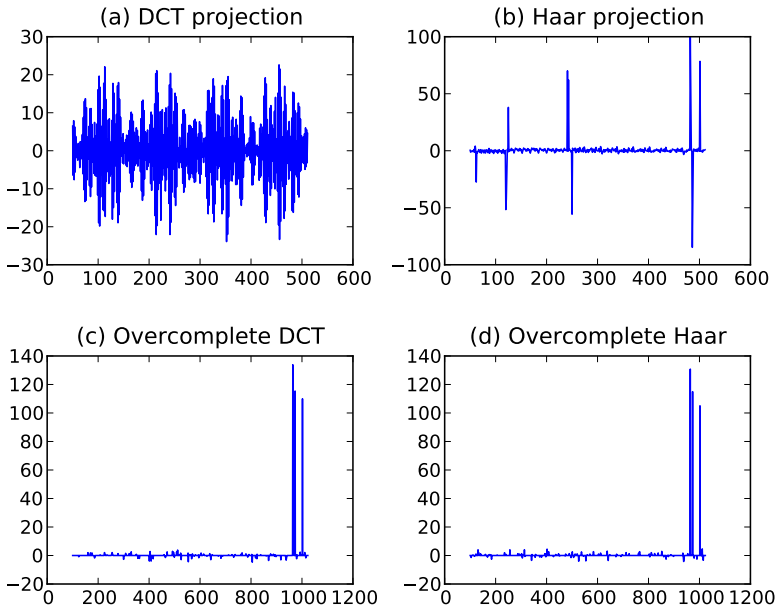


Figure 5.7: Compressive projection of the spatiotemporal signal contaminated with abnormal readings

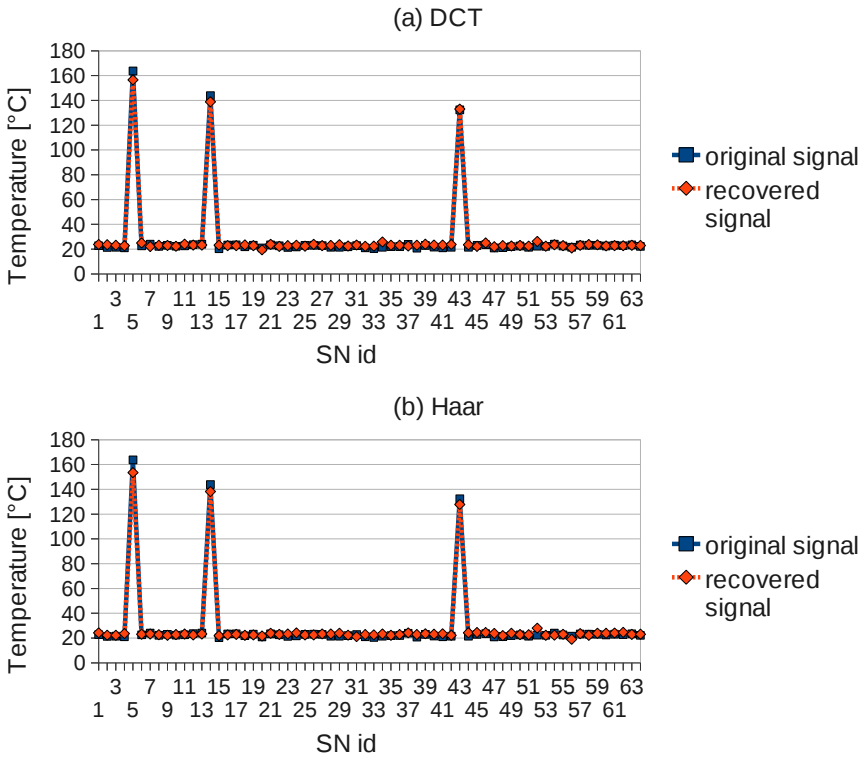


Figure 5.8: Event detection in multi-hop spatiotemporal CS using block-diagonal measurement matrix and overcomplete compressive systems

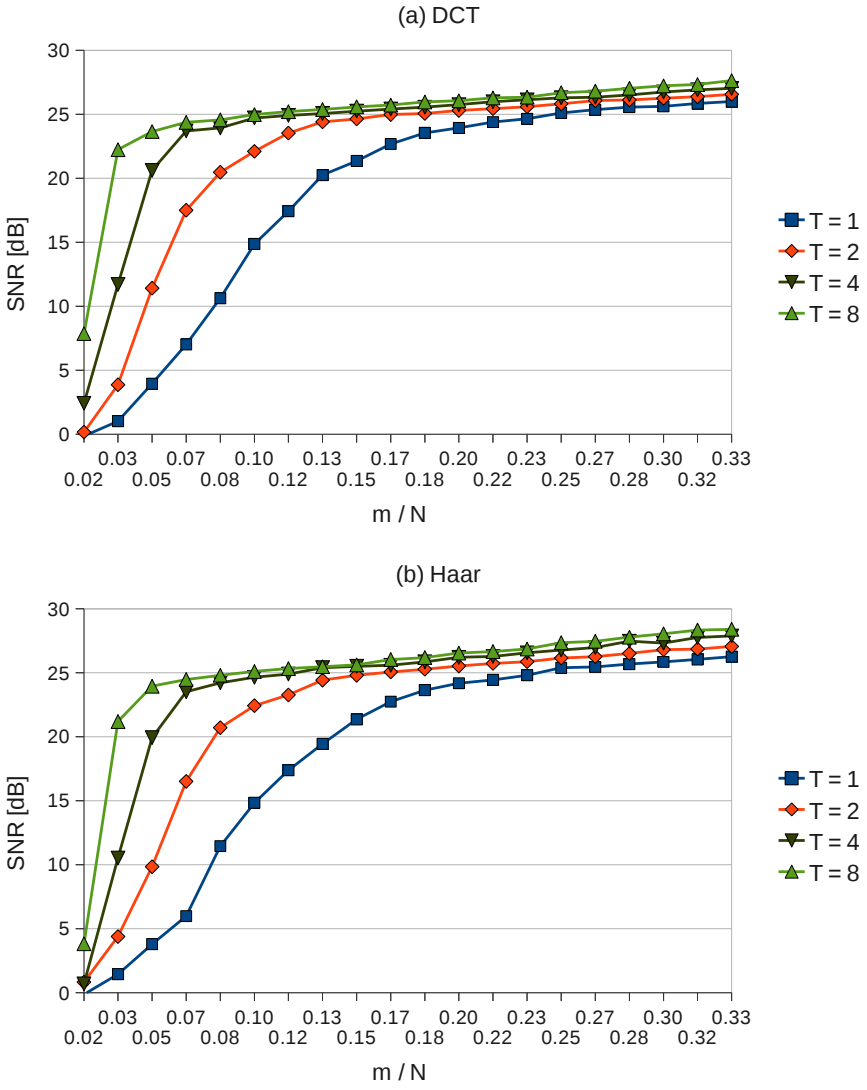


Figure 5.9: Reconstruction from DCS measurements

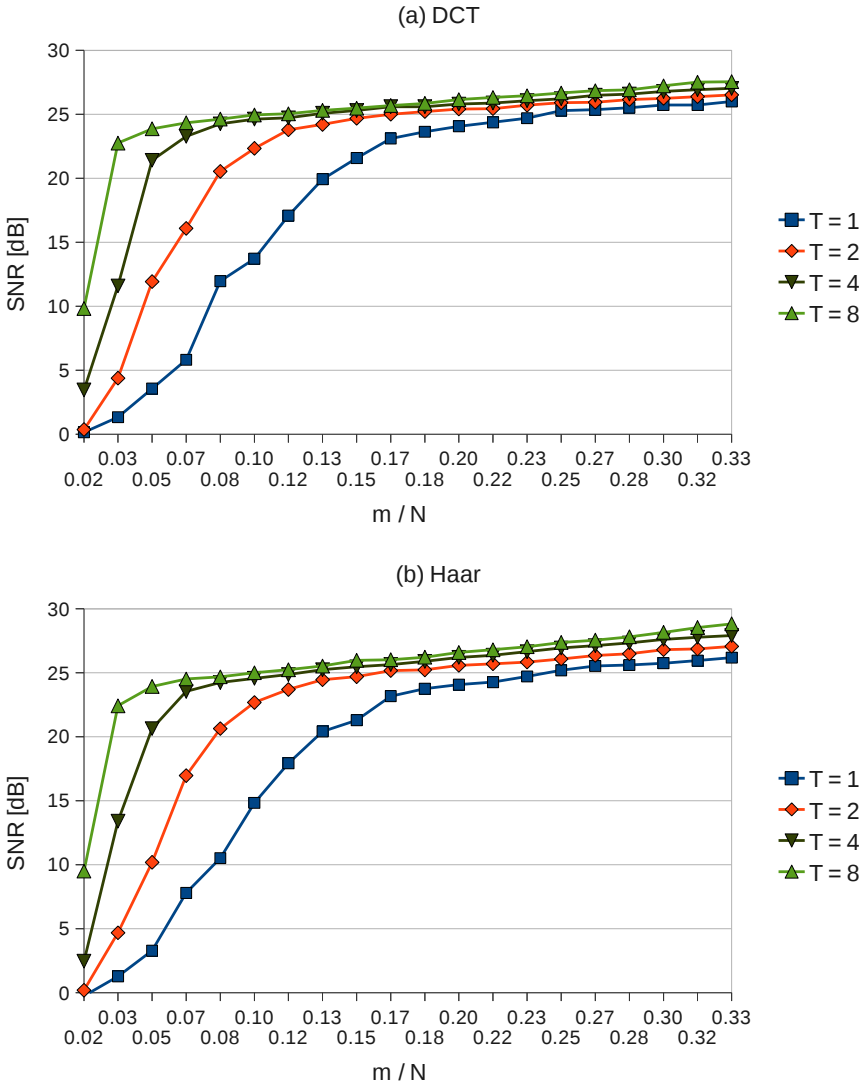


Figure 5.10: Signal recovery using dense Gaussian measurement matrix

more efficient, because the measurement cost of CS grows logarithmically by the size of the signal. Our evaluations on a real-world data set verifies our findings and shows better compressibility as well as the ability to detect abnormal events.

Chapter 6

Handling node and link failures

WSNs are often self-configured networks and their topology depends very much on the deployment and application requirements. In a two dimensional field such as a farm or a woodland, often a tree or chain topology is preferred for data collection. In this chapter, we target an important application of the WSNs, i.e, monitoring and surveillance of civil structures Chintalapudi et al. [2006]. In particular, we study WSNs with linear topology that is often employed for surveillance and monitoring of constructions such as roads, railways, bridges, etc. We say that a WSN possess a *linear* or *chain* topology, when the SNs are connected in a series and transmit the data hop by hop to deliver to the sink that is positioned at the end of this chain. Larger deployments may require several segments of chain WSNs with a sink at the end of each segment.

In a computation, power and bandwidth constrained WSN environment, the challenge is to efficiently transfer a large amount of the sensed data over a low-bandwidth wireless communication channel to the base station or sink. The sink is a dedicated node that receives the data from the Sensor Nodes (SNs) and prepares them for the end user. Since the SNs are battery powered, it is required to transmit the least amount of data in order to preserve more battery and prolong the lifetime of the WSN.

The problem of efficient data gathering is especially challenging in a WSN with linear topology. Figure 6.1 illustrates the baseline approach for data collection in a WSN with linear topology. The value sensed by SN i is denoted by f_i. Since the communication is done hop-by-hop, each node has to transmit its own data and also forward the data from the previous nodes. SN i has to transmit f_i and forward the values $\{f_1, f_2, \ldots, f_{i-1}\}$. Consequently, the nodes closer to the sink become highly overloaded. An effective solution to this problem is to reduce the amount of the transmissions by compressing the sensed data. Several studies show that, the data recorded by the SNs are

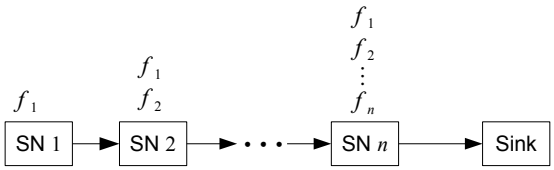


Figure 6.1: Baseline data transmission in a WSN with linear topology

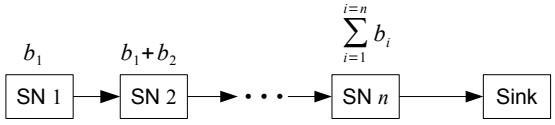


Figure 6.2: CS-based data collection in a WSN with linear topology

highly compressible Marco F. Duarte and Baraniuk [2012]; Srisooksai et al. [2012]. Thus, the use of compression algorithms such as CS to reduce the amount of data sent to the sink is advocated. An important requirement for the compression algorithm that is fulfilled by CS is to be light-weight as it runs on the resource-limited SN hardware platform Polastre et al. [2005].

6.1 CS in WSNs with linear topology

Figure 6.2 shows the CS-based data collection in a WSN with linear topology. It is a simplistic illustration of the Compressive Data Gathering (CDG) method that we explained in detail in Section 3.3.2. Each node computes an encoding of its sensed value. The encoded value by SN i is denoted by b_i. Since these values are *accumulated* (arithmetically added) at each hop, the amount of transmissions at all SNs remains equal. Thus, the communication and computation load is balanced across the network. This effectively avoids occurrence of exhausted nodes.

As we have seen in Chapter 3, measurement and recovery in CS is resilient to additive noise in the communication channel. This feature significantly distinguishes CS from other compression techniques. In traditional compression techniques, the compressed data are typically very sensitive to network perturbations Luo et al. [2009]. Therefore, additional protective techniques (e.g. Forward Error Correction (FEC) or Automatic Repeat reQuest (ARQ)) are employed to ensure reliable transfer of the data. In CS, additive noise is handled gracefully without putting more communication and controlling overhead Candes et al. [2006a]; Haupt and Nowak [2006].

While CS is shown to inherently handle additive noise without signifi-

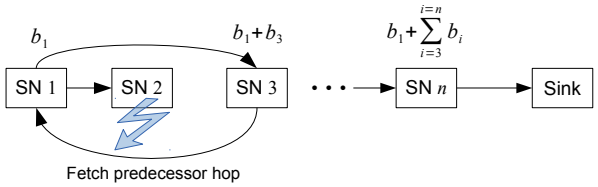


Figure 6.3: Node failure and chain reconstruction

cant communication or control overhead, CS's ability to withstand the very commonly encountered WSN failure occurrences of SNs or communication failure has yet to be explored. Addressing these node/link perturbations in CS is the specific objective of this chapter. We study the performance of CS under failure circumstances and perturbation models beyond mere additive noise that commonly occur in WSNs, but not considered seriously in the CS literature.

WSNs are usually deployed in uncontrolled operational environments, and hence, it can happen that some SNs get damaged and cannot continue their function. We study the problem of CS-based data collection in a chain WSN in presence of node failures. For simplicity of reading, we may use the term WSN or chain to refer to a WSN with chain or linear topology. When a node encounters a failure, it stops accumulating and forwarding the measurements to the next hop. This failure causes a breakage of the chain at the position of the failing SN.

6.1.1 Failures in a WSN with chain topology

Figure 6.3 illustrates a failure scenario in which SN 2 stops sending to the next hop. We assume that SN 3 detects this failure since it does not receive any messages from SN 2. Consequently, SN 3 fetches the measurement from the first healthy predecessor node, i.e., SN 1. Looking at the accumulated measurement that is received by the sink, we observe that only maintaining the chain connectivity is not sufficient to cancel the effect of node failure. The value of b_2 is missing in the accumulated measurement that is received by the sink. Consequently, the sink also has to exclude the missing samples when it wants to recover the original data. Therefore, a list of failed SNs must be communicated to the sink which in turn requires more communication overhead. Moreover, maintaining the consistency of such a list is a cumbersome task especially in a large-scale WSN.

CS is based on the fact that the data being sensed is compressible. For example, in WSNs, we know that the values sensed by the SNs have spatial

and temporal compressibility. Therefore, instead of transmitting the entire raw data to the sink, we send a small number of measurements. The sink can then recover the original data from the received measurements.

When a node or link failure in the WSN occurs, one or more sensed data items are missing or become corrupted. In this case, that part of the sensed data which is affected by the failure is not correlated to the rest of the sensed data anymore. Therefore, CS either fails or its performance degrades drastically. One possible solution is to exclude the faulty nodes from the measurement process.

This requires the sink to be informed exactly which part of the sensed data is excluded from compression. However, sending failure information to the sink requires more communication overhead. As we mentioned earlier, the set of raw data on which the compression takes place must be fixed during the compression and communication process. This means that it is not possible to add or remove arbitrarily to the raw data, otherwise the data correlation cannot be guaranteed.

We provide a solution for robust data gathering in WSNs using an enhanced version of CS that is capable of:

- Maintaining the connectivity of a chain WSN that performs CS-based data collection by making auxiliary links and isolating the failing nodes.
- Detecting the location of the failures without modifying the measurement mechanism at the SN level and without sending health-monitoring messages to the sink.
- Minimizing the effect of faulty or missing sensor readings on accuracy of the CS recovery algorithm.

We present a simple and effective enhancement to CS-based data gathering for WSNs with chain topology that is resilient to node and link failures as well as communication noise. In case of burst failure, e.g., when a node goes off-line for a period of time or in some cases forever, more and more retransmissions will only deplete the battery of the SN without any success in error recovery. Reconstruction of the chain is also a costly process. In such cases, the second approach is applied and the location of the failure is detected by the sink using our modified recovery algorithm. It is then possible to exclude the missing data and rerun the recovery algorithm to acquire the genuine data that are not affected by the failure.

Our evaluations show that the same technique for failure detection and isolation can be also applied to WSN with start topology without significant changes to the measurement and signal reconstruction mechanism.

6.2 CDG in chain topology

As discussed in Section 3.3.2, one advantage of CDG over CWS is that it can be implemented on the existing hardware platforms of the SNs. Instead of analogue communication and signal superposition, CDG uses established digital wireless communication standards such as ZigBee to exchange messages and transmit the measurements in the network Caione et al. [2010].

Furthermore, while CWS is suitable for star topology, CDG runs more efficiently on chain topologies as depicted in Figure 6.2. Given the measurement matrix $\mathbf{\Phi}$ as

$$\mathbf{\Phi} = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,n} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,n} \\ \vdots & & \ddots & \vdots \\ \phi_{m,1} & \phi_{m,2} & \cdots & \phi_{m,n} \end{pmatrix} \tag{6.1}$$

we define the column vector $\boldsymbol{\alpha}_i$ as $\boldsymbol{\alpha}_i = [\phi_{1,i}\ \phi_{2,i}\ \ldots\ \phi_{m,i}]^T$.

Each SN is given a unique id and runs a pseudo-random number generator algorithm seeded by its id to produce $\boldsymbol{\alpha}_i$. All of the SNs run the same pseudo-random number generator algorithm, though with different seeds. The pseudo-random number generator should not generate the same sequence of random values for two different SNs.

SN i senses the value f_i and multiplies this real number by the column vector $\boldsymbol{\alpha}_i$. If applicable, it accumulates the measurements received from the previous hop and forwards the result to the next hop. The same process is repeated by every SN till the measurement vector $\mathbf{y}$ is delivered to the sink, see Figure 6.4.

The measurement matrix $\mathbf{\Phi}$ can be easily reproduced at the sink by executing the same pseudo-random number generator seeded by the SN id's. Therefore, $\mathbf{\Phi}$ does not need to be communicated between the SNs and the sink. Having $\mathbf{\Phi}$ and $\mathbf{y}$, the sink can recover $\mathbf{f}$ from $\mathbf{y}$ after performing the CS reconstruction step. This process forms the main building block of many distributed data gathering techniques based on CS Bajwa et al. [2006]; Luo et al. [2009]; Mahmudimanesh et al. [2012].

CDG also introduces a method to detect abnormal sensor readings and eliminate their unwanted effects. Similar to CDG, we also use a post-processing detection method that is executed on the sink.

As discussed in pervious chapters, CDG uses over-complete dictionaries in CS signal recovery to detect abnormal sensor readings that are caused by either faulty nodes or abnormal events. In other words, the SNs do not actively participate in notifying about an abnormal sensor reading or a failure.

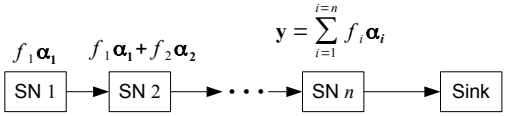


Figure 6.4: CDG in a WSN with linear topology

Instead, with the help of the recovery algorithm, the sink is able to detect whether there are faulty values in the received data. The difference of our work from CDG is that we consider the case of node failure in both sensing and forwarding roles. In the second case, we see that usually more than a few samples will be missed since a forwarding node carrying the payload from underlying levels may fail to transfer the measurements to the sink.

6.3 Handling node failures

In failure-free scenarios, the measurement vector $\mathbf{y}$ is received by the sink after running the network coding of CDG. When a node fails, what is received by the sink is different from what is expected to be received. As we show in Section 6.4, this causes anomalies in the signal that degrades the accuracy of the recovered signal.

The advantage of our work over CDG is that it handles node failures. By node failure, we mean the situation where a node gets damaged and cannot transmit or forward the measurements to the next hop. We propose a method to maintain the connectivity of the chain and rebuild the connection at the position of the failing node. Then we present our solution that applies a post-processing phase to detect the location of the failures and exclude the missing samples from the recovery process. If our connectivity restoring technique does not succeed due to a heavy damage to a burst of SNs, our detection method still precisely determines the location of chain breakage and also excludes the missing segment of the signal from signal recovery.

In Section 6.4, we see that if these failures are not detected, the accuracy of the reconstructed signal at the sink degrades significantly. Using our method, data collection continues seamlessly after rebuilding the chain and the missing samples are isolated to maintain the accuracy of signal recovery at the sink. This issue applies to both CWS and CDG as we will see in Section 6.4 where we describe our solution. We consider a WSN with linear topology consisting of static SNs and a single static sink at the end of the chain. The goal is to collect all of the sensed data at the sink.

6.3.1 Communication cost

The communication cost depends on both communication range and the number of messages. Sending more data consumes more battery power. Also achieving a more distant receiver requires to transmit with higher radio power. While the energy consumption grows linearly with the size of the transmitted data, it grows quadratically with the communication range. Total communication cost to transmit m messages to a receiver in the distance of d is $O(md^2)$.

For simplicity we assume that the nodes are placed on equal distances of each other. For a chain WSN consisting of n nodes, the distance between each two consecutive nodes is the same and equal to d. Let P_0 be the radio power required for communicating a unit of data between nodes i and $i+1$. Thus the radio power required for communication between nodes i and $i+2$ is $4P_0$, and in general, the radio power for communication between nodes i and $i+k$ is k^2P_0.

Node failure must be detectable by other nodes in order to rebuild the chain connectivity. Based on the detection method, we study three different approaches to maintain the connectivity of the chain.

- **AR** - Retransmission based on Acknowledgements: Each hop of the chain must send an acknowledgement when it successfully processes and forwards a measurement to the next hop. In case of a failure, no acknowledgement is sent and the previous hop retransmits the packet with higher radio power to reach the next functioning node.

- **IA** - Retransmission based on Implicit Acknowledgements: Instead of sending acknowledgements, each hop implicitly monitors the communication channel. When it detects that its next hop did not forward a measurement, it retransmits the measurement with higher radio power to reach the next healthy node.

- **MC** - Multiply-Connected chain: No explicit or implicit acknowledgement are used in this method. The nodes transmit all the time with a higher radio power and transmit their measurements to multiple next hops. Failures are recovered by each of the functioning nodes in the consecutive set of multiply linked chain.

In all of the above strategies, anomalies in the signal due to missing samples should be handled in the recovery process, otherwise the accuracy of the genuine data will be affected by the missing or corrupted data.

6.3.2 Sensor validation criteria

It is required that the range of valid sensor readings are known. We require that a SN calculates and transmits the measurements only when the sensor reading is within a finite range. For example, when a temperature sensor which is designed to measure values between -50 degrees to +500 degrees Celsius reports a value of -1000 or +2000, then the SN regards this value as invalid and does not compute the measurement, and consequently, no message will be sent by this SN.

Note that the term *measurement* is formally defined in Definition 5. It differs from sample or the value recorded by the sensor.

If the range of floating point numbers that a SN can store in its memory is $[B_L, B_U]$ and the range of valid sensor values is $[s_l, s_u]$, we require that

$$|s_l - s_u| \ll |B_L - B_U|. \tag{6.2}$$

The range of valid sensor readings must be bounded and the length of this range must be a small fraction of the range of numbers that can be stored in the memory of the SN. For example, a numerical storage format such as IEEE 754 half precision or single precision is suitable for a temperature sensor that records values e.g. between -50 and +500 degrees.

6.3.3 Scope of applications

In the simplest form, we consider that the SNs are solely performing the simplistic network coding to compute the measurements. The SNs are not involved in failure mitigation. In case of a failure, the neighboring nodes do not try to recover by employing methods such as retransmission or topology reconfiguration. All of the failure handling is done at the sink.

Our system model is particularly suitable for two classes of applications:

- WSN consisting of SNs with very basic hardware platforms in which we want the embedded software to be as simple as possible Kahn et al. [1999].
- Real time WSNs in which we want to gather the data within a specified deadline Stankovic et al. [2003]. In real time applications, non-deterministic delays caused by retransmissions should be avoided.

In a more complex scenario, we also consider employing retransmissions when the aggregated data are close to the sink. SNs closer to the sink usually carry the measurements that are aggregated over many lower-level SNs. Our

method in its simplest form can still handle failures at these nodes. However, a combination of retransmission and our post-processing failure handling leads to better overall performance.

6.4 Detecting and isolating failures

In this section, we describe our connectivity maintenance and failure detection technique in three steps using an illustrative example. First, we describe our technique for restoring the connectivity of the WSN when one or more nodes fail in the network. Second, we propose a method that detects the exact location of failing nodes without transmitting any health monitoring messages. Finally, we show that our failure detection technique withstands the extreme failures in which the connectivity of the chain is not recoverable.

6.4.1 Restoring connectivity in chain topology

In a normal operation of the WSN, all nodes are accumulating and transmitting their measurements hop by hop to deliver the measurement vector $\mathbf{y}$ to the sink. As described in Section 6.3.1, the nodes are regularly located in a series with distance d from each other. All nodes are transmitting with the radio power P_0 to communicate with their direct neighboring nodes.

All nodes are placed in a series arranged from node 1 to n as depicted in Figure 6.4. When node $i \in \{1, \ldots, n\}$ fails to transmit to node $i+1$, node $i+1$ detects this failure since it does not receive any messages from its previous hop. Consequently, it tries to contact the node $i-1$ and fetch its measurements. According to our system model, this requires $4P_0$ radio power since the nodes $i-1$ and $i+1$ are placed in a distance of $2d$.

Definition 12. Step-back count *is defined as the number k when node $i+1$ successfully fetches the measurements from node $i-k$ in case that the nodes $i-k+1, i-k+2, \ldots, i$ fail to deliver their encoded values to node $i+1$.*

A step-back of size k requires the nodes $i-k$ and $i+1$ to transmit with radio power $(k+1)^2 P_0$. Note that node $i+1$ sequentially tries to fetch data from nodes $i-1, \ldots, i-k+1$ until it reaches the first healthy predecessor $i-k$. Respectively, these trials has a cost of $4P_0, 9P_0, \ldots, k^2 P_0$ before reaching the healthy node $i-k$.

The maximum allowed step-back count is obviously not unlimited. Depending on the capabilities of SN's radio module, there is a limit for the maximum communication range.

Definition 13. Step-back limit k_{max} *is the maximum step-back count* k *that is allowed by the communication capabilities of the radio module of a sensor node.*

Depending on the success of the connectivity maintenance phase, one of the following cases may occur:

- **Successful network restoration**: The information flow continues by stepping back by $k \leq k_{max}$ hops. In this case, the connectivity of network is restored, though the samples $i-k+1, i-k+2, \ldots, i$ are missing due to node failures.

- **Unrecoverable chain breakage**: Restoration mechanism cannot rebuild the chain because even the node $i - k_{max}$ does not respond to the measurement fetching request that is sent by node $i+1$. All of the samples $1, 2, \ldots, i$ will be missed because of chain breakage.

We should emphasize that failure detection and recovery mechanism is entirely done at the sink by post-processing. The SNs do not actively participate in reporting which nodes have failed and which nodes are transmitting genuine data. Our technique avoids overhead of transmitting health monitoring messages and does not require acknowledgement or retransmission mechanism, thus, effectively reduces overhead of failure reporting.

6.4.2 Degrading effect of the missing samples

Here, we consider an illustrative synthesized spatial temperature signal that is compressible under DCT. Our discussion can easily extend to sensing any other physical parameter that is compressible in some compressive basis. Consider a WSN consisting of temperature sensors with linear topology consisting of $n = 256$ SNs. The values of the samples sensed by the SNs are represented as a spatial signal vector $\mathbf{f}$ of size 256.

In failure-free operation of the WSN, all of the SNs are transmitting their $f_i\boldsymbol{\alpha}_i$, and thus, the measurement vector $\mathbf{y}$ is correctly received by the sink. Suppose that when none of the samples of $\mathbf{f}$ are missing, the vector $\mathbf{f}$ is compressible under DCT. More precisely, $\mathbf{f} = \boldsymbol{\Psi}_D\mathbf{x}$ where $\boldsymbol{\Psi}_D$ is the $n \times n$ inverse DCT matrix and $\mathbf{x}$ is sparse. We set the sparsity of $\mathbf{x}$ to 10 in the synthesized signal of our example. To make it more realistic, we add a white Gaussian noise to the measurement vector $\mathbf{y}$. We have set the power of the noise to be 5% of the signal power.

According to our setup, to estimate the original signal $\mathbf{f}$ from the measurement vector $\mathbf{y}$, we have to solve the following convex optimization problem.

$$\underset{\tilde{\mathbf{x}}\in\mathbb{R}^n}{\text{minimize}} \ \|\tilde{\mathbf{x}}\|_1 \ , \ \text{subject to} \ \|\mathbf{y} - \mathbf{\Phi}\mathbf{\Psi}_D\tilde{\mathbf{x}}\|_2^2 \ \leq \ \epsilon \tag{6.3}$$

If $\hat{\mathbf{x}}$ is the solution to Equation 6.3, then the original signal is estimated by $\hat{\mathbf{f}} = \mathbf{\Psi}_D\hat{\mathbf{x}}$.

In the failure-free case, the recovered signal $\hat{\mathbf{f}}$ shows a good accuracy compared to the original signal $\mathbf{f}$, see Figure 6.5.a. When the connectivity of the chain is maintained, node failures are equivalent to missing samples in the signal vector $\mathbf{f}$. When some of the samples are missing due to node failures, the corresponding elements of vector $\mathbf{f}$ suddenly drop to zero. When this happens, there is no guarantee that $\mathbf{f}$ is compressible under DCT anymore, and hence, we cannot accurately recover the original vector $\mathbf{f}$ by solving Equation 6.3. Looking at Figure 6.5.b we observe that the accuracy of signal recovery significantly decreases. The recovery of the latter case shows a significant accuracy loss as illustrated in Figure 6.5.b compared to failure-free case depicted in Figure 6.5.a. The Signal-to-Noise Ratio (SNR) drops from around 65 to 19 when some of the samples are missing due to SN failures. Furthermore, it is not possible to detect which nodes are failing.

Next, we present a method that exactly detects the location of failures and excludes the failing SNs from the signal recovery process.

6.5 Signal elevation during measurement

From our system model description we recall that the values recorded by the SNs are bounded between a lower and upper bound, namely s_l and s_u respectively. In our exemplified WSN, we assume that the temperature values are bounded between 0 and 25. Note that s_l and s_u can be any positive or negative real numbers. We take 0 and 25 just as an example here. This discussion also applies to signals recorded from any other physical parameter other than temperature.

Suppose that each SN *elevates* its recorded value by an *offset* before applying the measurement mechanism illustrated in Figure 6.4. Formally speaking, when SN i senses the value f_i, it first adds f_i by a positive real number c that we call it *offset* and then multiplies $\boldsymbol{\alpha}_i$ by $f_i + c$.

We choose c to be two or three order of magnitudes larger than $|s_l - s_u|$. The reason is that when node failures occur, the missing samples are better distinguished from the genuine samples. In fact, the offset elevates the range of valid values to a higher level. In our current example, if we select $c = 1000$, then the range of offsetted values will be $[1000, 1025]$ instead of $[0, 25]$. Using this simple modification in the measurement mechanism, we distinguish a

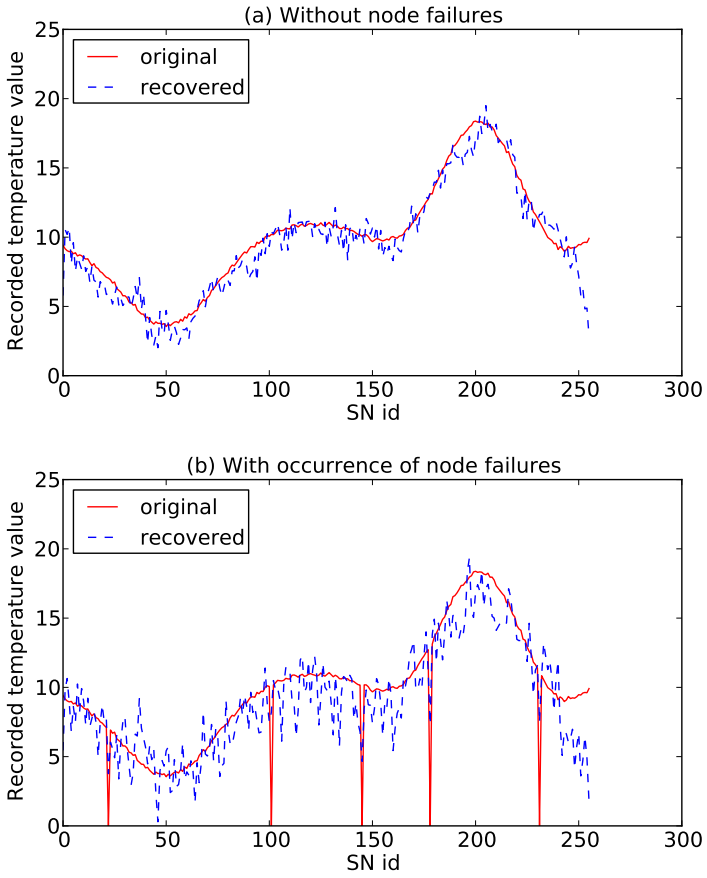


Figure 6.5: Degraded signal recovery due to missing samples

reported value zero due to missing samples from the case when the recorded value is in fact zero.

Note that when all nodes perform the offsetting, the whole signal vector $\mathbf{f}$ is elevated by the offset c. Let $\mathbf{g} \in \mathbb{R}^n$ be the elevated version of $\mathbf{f}$, i.e.,

$$\mathbf{g} = \mathbf{f} + \mathbf{c}_n \tag{6.4}$$

where $\mathbf{c}_n$ is a column vector of size n with all of its elements being equal to c.

6.5.1 Detection and exclusion of the missing samples

When some nodes fail, their corresponding values in $\mathbf{g}$ will also drop to zero, since they cannot transmit their measurements. In this case, the compressibility of $\mathbf{f}$ (and also $\mathbf{g}$) under DCT decreases. However, $\mathbf{g}$ shows high compressibility under Haar wavelet transform Graps [1995]. In order to find out which nodes are missing, we try to recover $\mathbf{g}$ by solving the following convex optimization problem.

$$\underset{\tilde{\mathbf{u}} \in \mathbb{R}^n}{\text{minimize}} \ \|\tilde{\mathbf{u}}\|_1 \quad , \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{\Phi}\mathbf{\Psi}_H \tilde{\mathbf{u}}\|_2^2 \ \leq \ \epsilon \tag{6.5}$$

where $\mathbf{\Psi}_H$ is the inverse Haar transformation matrix.

If $\hat{\mathbf{u}}$ is the solution to Equation 6.5, then $\hat{\mathbf{g}} = \mathbf{\Psi}_H \hat{\mathbf{u}}$ estimates the original elevated signal $\mathbf{g}$. The size of the signal must be a power of two when applying the Haar transform. In this example, the size of the signal is equal to the number of SNs, i.e., $n = 256$. When the number of nodes is not a power of two, one can pad sufficient number of pseudo-samples with a predetermined value. Pseudo-samples with a value of c is a suitable choice here.

Recovery result using Haar wavelet is depicted in Figure 6.6.a. Here, $s_l = 0$, $s_u = 25$ and $c = 1000$. While recovery using DCT cannot distinguish the failing SNs, using Haar wavelet as our compressive basis, we can exactly detect the location of the failing nodes. The estimated signal $\hat{\mathbf{g}}$ recovered by solving Equation 6.5 shows significantly lower values at the failing nodes, see Figure 6.6.a. By employing signal elevation and performing the reconstruction using Haar wavelet, the exact location of the failures in a chain WSN is determined.

After detecting the position of the missing samples, we exclude those samples and rerun the signal recovery on that part of the signal that actually contains valid sensor readings.

Using matrix representations, the measurement vector received by the sink is given by

$$\mathbf{y} = \mathbf{\Phi}\mathbf{g} + \mathbf{z} \tag{6.6}$$

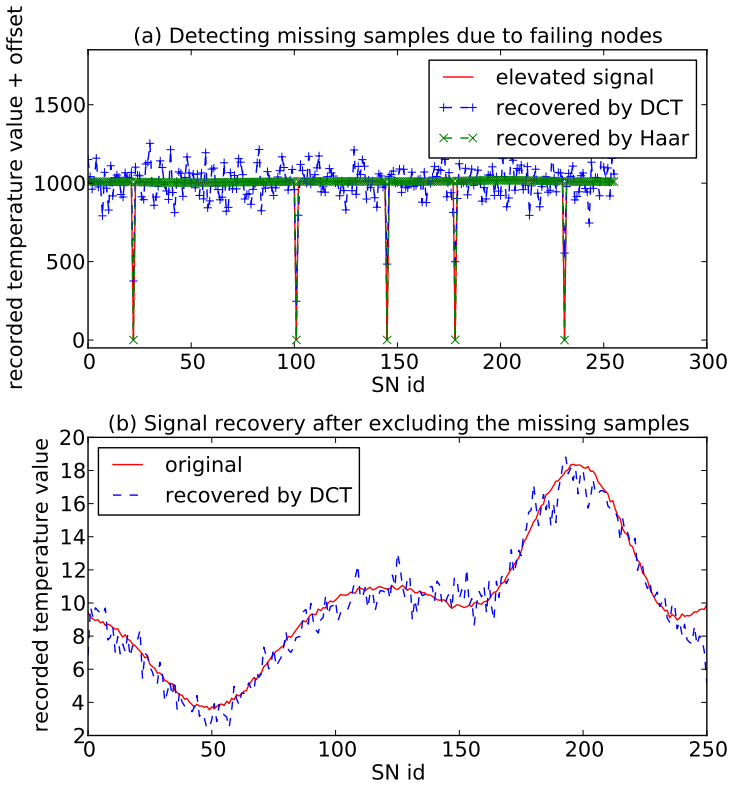


Figure 6.6: Detecting and excluding the failing SNs

where $\mathbf{z}$ is the additive noise. We add a white Gaussian noise with a power equal to 5% of the signal power in all of our simulations. This resembles the communication noise or computation precision noise of the SNs.

Let M be the set of failing nodes and k be the number of failing nodes, i.e., $k = |M|$. We define an $m \times (n-k)$ matrix $\mathbf{\Phi'}$ by removing the columns $m_1, m_2, \ldots, m_k$ from $\mathbf{\Phi}$ where $\{m_1, m_2, \ldots, m_k\} = M$.

$$\mathbf{\Phi'} := [\Phi_{i,j}] \quad i \in \{1, \ldots, m\} \ , \ j \in \{1, \ldots, n\} - M \tag{6.7}$$

We define the vector $\mathbf{g'}$ of size $(n-k)$ by removing the elements g_i from $\mathbf{g}$ where $i \in M$.

$$\mathbf{g'} := [g_i] \quad i \in \{1, \ldots, n\} - M \tag{6.8}$$

It is easy to show that:

$$\mathbf{\Phi g} = \mathbf{\Phi' g'}. \tag{6.9}$$

We also define the column vector $\mathbf{f'}$ by removing the elements f_i from $\mathbf{f}$ where $i \in M$.

$$\mathbf{f'} := [f_i] \quad i \in \{1, \ldots, n\} - M \tag{6.10}$$

According to the definitions of $\mathbf{g}$ and $\mathbf{g'}$, we know that:

$$\mathbf{g'} = \mathbf{f'} + \mathbf{c}_{(n-k)} \tag{6.11}$$

where $\mathbf{c}_{(n-k)}$ is a column vector of size $(n-k)$ with all of its elements being equal to c.

Since $\mathbf{f}$ is compressible under DCT and $\mathbf{f'}$ is equal to $\mathbf{f}$ except some missing samples, it is still expected to be sufficiently compressible under DCT, i.e.,

$$\mathbf{f'} = \mathbf{\Psi'}_D \mathbf{x'} \tag{6.12}$$

such that $\mathbf{\Psi'}_D$ is the $(n-k) \times (n-k)$ inverse DCT matrix and $\mathbf{x'} \in \mathbb{R}^{(n-k)}$ is the (nearly) sparse projection of $\mathbf{f'}$ under DCT. Note that any compressive basic other than DCT can also apply here. Choosing DCT is independent of our offsetting and detection mechanism using Haar wavelets. We use DCT as a an example and this discussion can be generalized to any other compressive basis Ψ when $\mathbf{f} = \mathbf{\Psi x}$ and $\mathbf{x}$ is sparse or nearly sparse.

Putting Equation 6.9 in Equation 6.6 we have:

$$\mathbf{y} = \mathbf{\Phi' g'} + \mathbf{z} \tag{6.13}$$

and by substituting Equation 6.11 we have:

$$\begin{aligned} \mathbf{y} &= \mathbf{\Phi'}(\mathbf{f'} + \mathbf{c}_{(n-k)}) + \mathbf{z} \\ &= \mathbf{\Phi' f'} + \mathbf{\Phi' c}_{(n-k)} + \mathbf{z} \end{aligned} \tag{6.14}$$

We define a vector $\mathbf{w} \in \mathbb{R}^m$ as:

$$\mathbf{w} := \mathbf{y} - \mathbf{\Phi}'\mathbf{c}_{(n-k)} \tag{6.15}$$

According to the definition of $\mathbf{w}$ and Equation 6.14, we have:

$$\mathbf{w} = \mathbf{\Phi}'\mathbf{f}' + \mathbf{z}. \tag{6.16}$$

Therefore, we can recover $\mathbf{x}'$ by solving a modified version of Equation 7.5 as follows.

$$\underset{\tilde{\mathbf{x}}' \in \mathbb{R}^n}{\text{minimize}} \ \|\tilde{\mathbf{x}}'\|_1 \quad , \quad \text{subject to} \quad \|\mathbf{w} - \mathbf{\Phi}'\mathbf{\Psi}'_D\tilde{\mathbf{x}}'\|_2^2 \ \leq \ \epsilon \tag{6.17}$$

If $\hat{\mathbf{x}}'$ is the solution to Equation 6.17, the original signal excluding the missing samples, i.e., $\mathbf{f}'$ is then estimated by $\hat{\mathbf{f}}' = \mathbf{\Psi}'_D\hat{\mathbf{x}}'$. The recovery result is shown in Figure 6.6.b for our current example.

By comparing Figures 6.6.b and 6.5.b to each other, we see that, a much more accurate signal recovery is possible after excluding the missing samples. Figure 6.5.b shows the recovery when some of the samples are missing due to node failures. Figure 6.6.a shows how our technique can exactly detect the samples that are lost due to failing SNs. Figure 6.6.b shows the recovery of the same signal after excluding the missing samples. We observe a significantly more accurate signal reconstruction after exclusion of the missing samples.

Note that in Figure 6.6.b the size of the recovered signal is 251 instead of 256, because 5 samples corresponding to the failing SNs are excluded.

6.5.2 Detecting unrecoverable chain breakage

In this section we study the situation where the chains break at some node and our chain rebuilding procedure as described in Section 6.4.1 does not restore the connectivity. We do not want the next hops after the failing nodes to wait for the retransmission of the lost encoded values. In particular, when the failing nodes face an unrecoverable error, those encoded values may never be retransmitted and the next hop will wait forever for the missing part of the measurement vector. This situation is depicted in Figure 6.7.a. We let the next healthy node in the chain to send its own measurements without waiting for its failing predecessor nodes, see Figure 6.7.b. We apply the same technique as discussed in the previous section. The SNs elevate their sensed values by an offset c. Then, we first perform signal recovery using Haar wavelet to detect the position of the failure. Finally, we exclude the missing portion of the spatial signal and rerun the recovery on the valid part

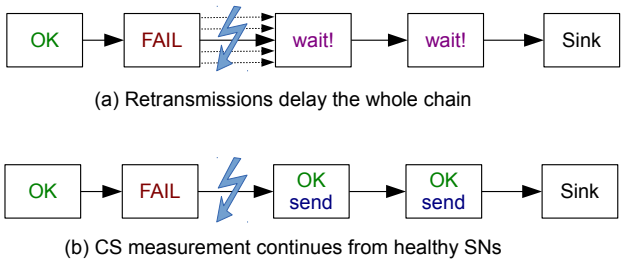


Figure 6.7: Post-processing failure detection without retransmission overhead

of the signal using the compressive basis under which the genuine data are compressible.

The conditions are the same as our example in the previous section. The signal $\mathbf{f}$ is compressible under DCT and records temperature values between 0 and 25 degrees which are elevated by $c = 1000$. We inject a failure in SNs $\{50 - k_{max}, \ldots, 50\}$. Thus, the sensed values from nodes $\{1, \ldots, 50\}$ will be lost as the chain restoration mechanism at node 50 is unsuccessful.

Recovery of the elevated signal after solving Equation 6.5 is depicted in Figure 6.8.a. We observe that recovery using DCT cannot exactly determine the location of the failing node. It gives only a rough estimation of the location where the chain is broken. On the other hand, recovery using Haar wavelet exactly distinguishes the missing segment of the signal and determines the position of the failure. We exclude the missing segment of the signal from recovery, here the samples 1-50 as depicted by the dashed curve in Figure 6.8.b.

Signal recovery using DCT without excluding the missing segment still gives us a good accuracy. However, it erroneously detects low temperatures for SNs 1-50, see the dotted curve in Figure 6.8.b. The recovered signal after excluding the missing segment retains its accuracy and also distinguishes the missing samples for nodes 1-50, see the dashed curve in Figure 6.8.b.

6.6 Evaluation

We have tested our technique on different simulated WSNs with our illustrative synthesized compressible signals. In this section, we apply our method on real-world datasets and put the WSN under stress tests to evaluate the performance of our failure detection technique.

The simulations are performed on a real-world dataset form the Sensorscope deployment LUC [2008]. Since not all of the SNs in the testbed

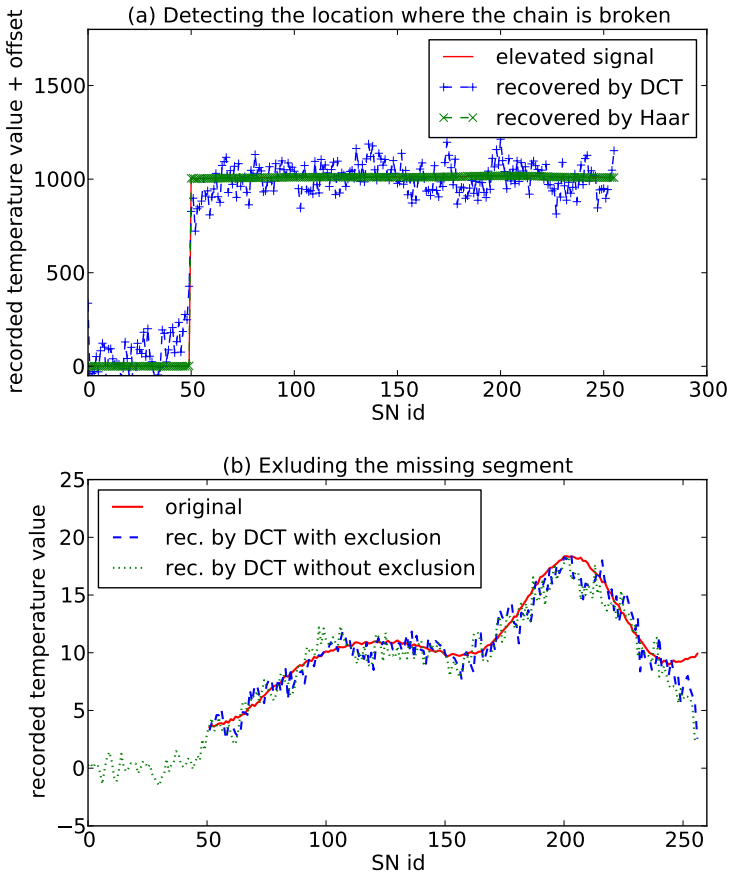


Figure 6.8: Detecting the location of unrecoverable chain breakage

were sampling synchronously, we selected 64 SNs with the most amount of synchronously sampled data. The sensed data from LUCE dataset is applied in our simulated network that possesses a linear topology. We take the data from the real-world dataset while the network topology is determined in our simulation program to be a linear topology. We employ our implementation of the recovery algorithm which uses CVXOPT software package Dahl and Vandenberghe [2006]. The simulation is implemented in Python using the NumPy/SciPy scientific programming libraries Jones et al. [2001–].

6.6.1 CDG in WSN with chain topology

First, we evaluate the effectiveness of our step-back method introduced in Section 6.4.1. In particular, we want to analyze the behavior of the step-back method for different values of k_{max} and different number of failure occurrences. We assume that each node has an independent probability p of failure. We simulate the chain WSN of size 64 with different values for $k_{max} \in \{1, 2, 3\}$ and different values of node failure probability $p \in [0.01, 0.15]$. Looking at Figure 6.9.a, we observe that the step-back method effectively reduces the amount of lost samples. Without the step-back method, i.e. when $k_{max} = 0$, we loose a lot of samples whenever a single node fails to transmit its values. This happens because when $k_{max} = 0$ and no step-back takes place, any node failures lead to a chain breakage. The step-back method tries to maintain the connectivity of the network when a node failure occurs. With higher k_{max} each node tries to fetch the values from farther predecessor nodes when their direct predecessor does not send them any data.

Observation 1. *The step-back method significantly reduces the lost samples when restoring the connectivity of the chain succeeds.*

Another interesting observation is that the average amount of increase in power consumption and the balance of load on the SNs mainly depends on the probability failure p. Figure 6.9.b illustrates the increase in communication cost and its standard deviation across the network after applying the step-back method for different values of p and k_{max}. We observe a moderate increase in average power consumption when applying the step-back method. The standard deviation shows how the load is distributed on the network. Higher standard deviation indicates that the nodes that are restoring the chain connectivity are overloaded.

Observation 2. *Higher step-back limit k_{max} increases the ability to restore the chain connectivity. Changes in power consumption of the WSN is prevalently determined by the failure probability p rather than step-back limit k_{max}.*

According to Observation 2 it is recommended to use a higher step-back limit k_{max} as long as the hardware capabilities of the SN allows it.

Now, we put the WSN under *stress test* and measure the accuracy of the recovered signal. The stress test deliberately fails some of the SNs, i.e., zeros their corresponding samples in the spatial signal. We measure the accuracy of signal recovery when the number of failures increases. The accuracy of the recovered signal is given by Signal-to-Noise Ratio (SNR) which is measured in decibels (dB). In our simulated network, setting $k_{max} = 5$ effectively maintains the connectivity of the chain. Figure 6.10 illustrates the signal reconstruction accuracy of our method and compares it to CDG for chain topology Luo et al. [2009]. Equipping CDG with out failure detection and isolation mechanism improves the accuracy of the recovered signal. Note that each unit dB increment of SNR roughly corresponds to 25% higher accuracy of the reconstructed signal as we measure the SNR by a logarithmic scale.

Observation 3. *Detection and isolation of the missing samples improves the accuracy of the recovered signal in Compressive Data Gathering.*

The simulations are executed on the real-world data from the Sensorscope dataset, while the network topology is controlled by the simulation environment. To assess failure detection capability in CWS, we apply a star topology. For CDG we run the simulations on a large set of randomly generated trees. This gives us a better understanding of the performance of our failure detection technique for a variety of WSN configurations, while the real-world data resembles a more realistic situation. Similar to our previous experiments, the measurements are contaminated with Gaussian noise. The power of the noise is 5% of the power of the signal.

6.6.2 CWS in star topology

Figure 6.11 compares the performance of conventional CWS with our enhanced version that is equipped with failure detection. We observe that by increasing the number of failures, the accuracy of the signal recovered by conventional CWS decreases steadily. Detection and exclusion of the missing samples keeps the accuracy at a higher level even when more failures are injected. We have inserted the failures in random locations and ran the simulation for 50 different time spots of the dataset. The simulation results are then averaged over 10 repetitions. Note that we assess both conventional CWS and failure-aware CWS using a WSN with the same location of failures.

Our failure detection technique helps to withstand the negative effect of failing SNs by excluding the missing samples from signal recovery. The result

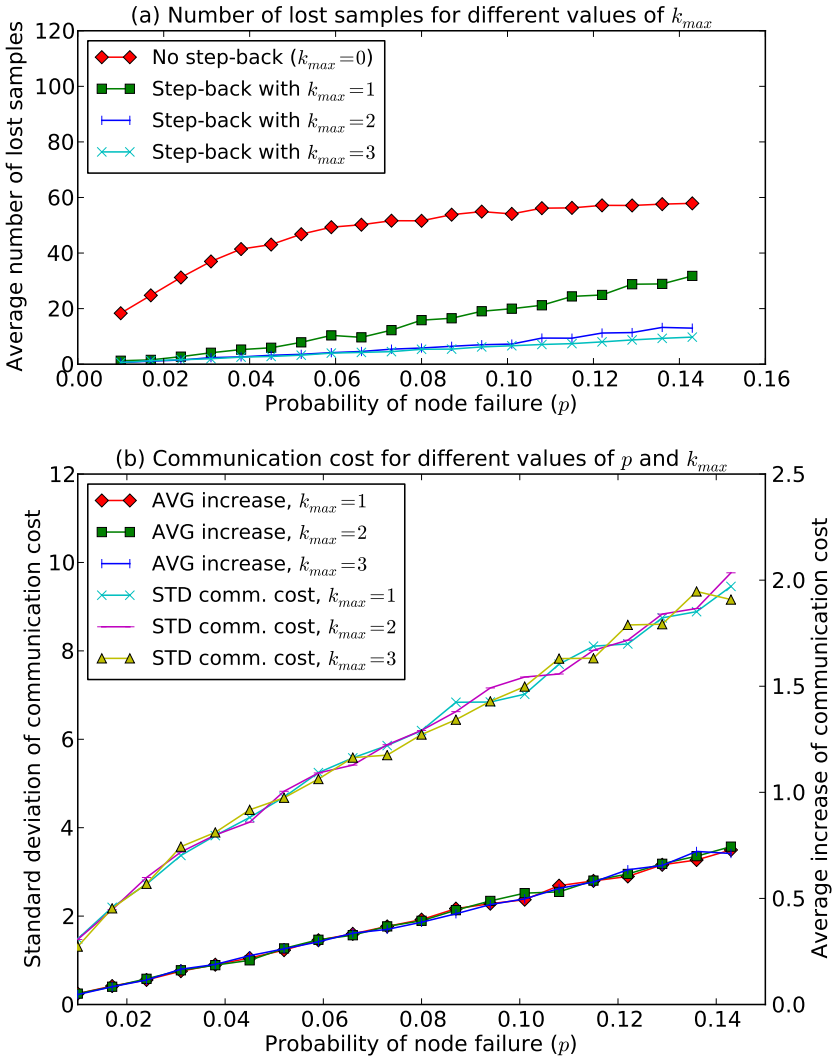


Figure 6.9: Analysis of the step-back method

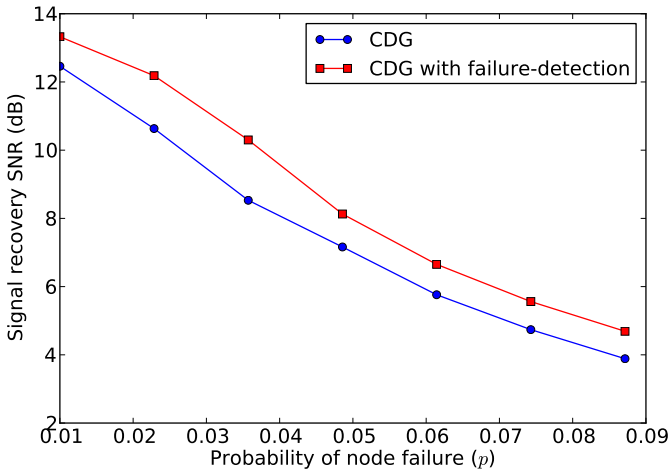


Figure 6.10: More accurate recovery after isolation of the missing samples

illustrated in Figure 6.11 is then averaged over many different simulation runs.

6.7 Summary

In this chapter, we introduced an enhancement to Compressive Data Gathering (CDG) in chain Wireless Sensor Networks (WSNs). CS-based data collection methods for WSNs like CDG are inherently robust to additive communication noise. In addition to communication noise, a WSN also faces another source of erroneous data collection. The WSNs are usually deployed in harsh operational environments, and thus, it is likely that some SNs gets damaged and cannot continue their function. Thus, the SNs of a WSN are at the risk of temporary or permanent defects.

The performance of CDG in WSNs with linear topology is also studied under circumstances where a SN encounters a failure and cannot transmit its measurements or forward the accumulated measurements from other SNs.

We proposed a simple and effective method based on a best-effort technique to maintain the connectivity of the chain topology. We also introduce an enhancement to CS measurement and recovery that excludes the missing samples due to node failures without transmitting health monitoring messages. Our proposed technique, first determines the location of the missing samples that are caused by node failures. Then, the recovery algorithm is executed on the remaining part of the signal that contains genuine data. Our

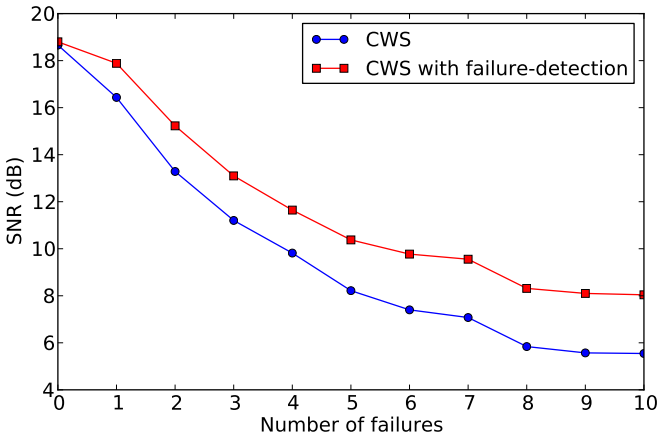


Figure 6.11: Comparing CWS with and without failure detection under stress test

evaluations prove that exclusion of the missing samples significantly improves the accuracy of the recovered signal.

Chapter 7

Data Dissemination via Network Coding

In a typical WSN, all of the sensed data is cooperatively gathered at the sink. The end user of the WSN fetches these data from the sink for further processing. In this chapter, we present an all-to-all dissemination method such that all of the sensed data is accessible from any node of a WSN. In simple words, each node can potentially be a sink. This grants more flexibility and mobility to the end user of the WSN, since it is possible to access the *global state* of the environment from any arbitrary SN in its vicinity.

Consider a WSN consisting of n static SNs. Suppose that vector $\mathbf{f} \in \mathbb{R}^n$ is made by stacking the values recorded by the SNs. Thus, the ith entry of vector $\mathbf{f}$, namely f_i, is equal to the value recorded by the ith SN. Like the previous chapters, we may use either of the terms spatial *signal* or *vector* interchangeably and both refer to $\mathbf{f}$. The goal of our dissemination method is to make the vector $\mathbf{f}$ available to all SNs within a certain time limit. This is useful for Wireless Sensor and Actuator Networks (WSANs) when a distributed control is performed based on the global state of the environment Verdone et al. [2008]. We also consider the scenario where the global state can be recovered by fetching a limited amount of data from a small subset of the SNs; e.g., when a mobile sink visits some SNs and estimates the global state of the environment by extracting their data.

There are two challenges to achieve this goal. First, the number of transmissions by the SNs must be minimized in order to meet bandwidth limitations and also save battery. Second, the dissemination protocol must be light-weight such that it can be easily implemented on the basic hardware platforms of the SNs.

In this chapter, we introduce *Comprensus*, a novel protocol for efficient dissemination of *compressible* data in a WSN. Comprensus draws its con-

cepts from the well known *consensus* methods studied in distributed control literature Olfati-Saber et al. [2006]. Unlike consensus, compressibility of the data plays a decisive role in our protocol. The more compressible the spatial signal is, the less transmissions are required by Comprensus to disseminate the data. Comprensus proves a significant performance gain over consensus techniques by exploiting the compressibility of the sensed data and reducing the number of transmissions.

Our approach has two advantages over the straightforward solution that gathers data at a stationary sink and sends it to the mobile end user. First, our method allows each SN to be a potential sink. Second, there is no need for the end user to be in radio range of the stationary sink. The mobile end user can extract the global state of the environment from any SN in its vicinity. This is especially useful for in-door applications where the base station is not necessarily accessible from any arbitrary location in the environment.

Evaluations show that using Comprensus each SN can have an estimation of the global state with a Signal-to-Noise Ratio (SNR) of more than 50 decibels. This level of SNR indicates that Comprensus can achieve a highly accurate estimation of the global state at all SNs. In Section 7.3 we show that under a fixed accuracy requirement, a tradeoff between latency and energy consumption can be settled depending on specific application requirements. We investigate scenarios ranging from low-latency energy-aggressive mode to energy-preserving high-latency mode. Comprensus proves to be easily tunable to each of these configurations.

Next, we briefly review the so called *RIPless* version of the CS theory and study its applications in WSNs. As we will see later in this chapter, the CS theory without the RIP (Restricted Isometry Property) allows us to define a custom measurement matrix that better suites our network coding technqiue.

7.1 RIPless Compressed Sensing

The CS theory is initially based on the Restricted Isometry Property (RIP) Candes [2008]. The Comprensus protocol is based on the newer version of the CS theory that does not require the RIP. The so called *RIPless* CS theory allows for a computationally feasible method to certify whether the preconditions of accurate signal recovery hold for a particular setup of Comprensus Candes and Plan [2011].

Definition 14. *We call a vector* $\boldsymbol{\phi} \in \mathbb{R}^n$ *a* sensing vector*, and the inner product of a sensing vector and vector* $\mathbf{f}$ *is called a* measurement*.*

Let $y_1, y_2, \ldots, y_m$ be m measurements such that

$$y_j = \boldsymbol{\phi}_j{}^T \mathbf{f} + z_j \quad , \quad j \in \{1, 2, \ldots, m\} \tag{7.1}$$

where $\boldsymbol{\phi}_j$ are the sensing vectors, $\{z_j\}$ is the white noise sequence with variance σ^2. This can be also written using matrix notations:

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{f} + \mathbf{z} \tag{7.2}$$

where $\mathbf{y} = [y_1 \ y_2 \ \ldots \ y_m]^T$ is the measurement vector, $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \ \ldots \ \boldsymbol{\phi}_m]^T$ is the measurement matrix and $\mathbf{z} = [z_1 \ z_2 \ \ldots \ z_m]^T$ is the noise vector.

According to the CS theory, it is possible to recover vector $\mathbf{f}$ from $m < n$ measurements under certain conditions for $\mathbf{f}$ and the sensing vectors as follows.

We assume that $\mathbf{f}$ can be sparsely represented under a linear projection using an orthonormal matrix $\boldsymbol{\Psi}$. Suppose that $\mathbf{f} = \boldsymbol{\Psi}\mathbf{x}$ for a suitably chosen orthonormal matrix $\boldsymbol{\Psi}$ such that $\mathbf{x}$ is sparse. Remeber that a vector $\mathbf{x}$ is called a sparse vector when it has $s \ll n$ non-zero components and all its other $(n-s)$ components are zero. As we have seen in the previous chapters, sparsity plays an important role in the CS theory. The sparser the vector $\mathbf{x}$ is, the fewer measurements are required to recover $\mathbf{f}$.

7.1.1 Isotropy and incoherence

Isotropy and *incoherence* are the other necessary conditions in order to recover $\mathbf{f}$ from $\mathbf{y}$ Candes and Plan [2011]. Let $\boldsymbol{\phi} \in \mathbb{R}^n$ be a random sensing vector with independent and identically distributed components drawn from distribution F, i.e., $\boldsymbol{\phi} \overset{iid}{\sim} F$.

Definition 15. *Distribution F has the* isotropy property*, when $\boldsymbol{\phi}\boldsymbol{\phi}^T$ is expected to be the identity matrix. Mathematically,*

$$E[\boldsymbol{\phi}\boldsymbol{\phi}^T] = \mathbf{I} \quad , \quad \boldsymbol{\phi} \sim F. \tag{7.3}$$

The isotropy condition can be weakened to *near isotropy*, i.e., $E[\boldsymbol{\phi}\boldsymbol{\phi}^T] \approx \mathbf{I}$ and still $\mathbf{f}$ is accurately recoverable from the measurement vector $\mathbf{y}$ Candes and Plan [2011].

Definition 16. Coherence parameter μ *is defined as the smallest value μ such that*

$$|\boldsymbol{\phi}_j^T \boldsymbol{\psi}_i|^2 \ \leq \ \mu \tag{7.4}$$

for all sensing vectors $\boldsymbol{\phi}_j$ and columns $\boldsymbol{\psi}_i$ of matrix $\boldsymbol{\Psi}$, $j \in \{1, 2, \ldots, m\}$ and $i \in \{1, 2, \ldots, n\}$.

We say that the sensing vectors are more *incoherent* if the value of μ is a smaller number. According to Candes and Plan [2011], the more incoherent the sensing vectors are, the less random measurements are required for accurate recovery. Candes et al. in their RIPless theory of CS Candes and Plan [2011] discuss some of the random distributions F obeying the isotropy condition. These include the Gaussian distribution, Rademacher distribution and random Fourier sampling Candes and Plan [2011]. Randomized sampling brings a key benefit for WSNs by eliminating the need for centralized coordination Bajwa et al. [2007]; Luo et al. [2009].

It is shown in Candes and Plan [2011] that if the isotropy condition holds and the number of measurements m is in the order of $O(\mu s \log n)$, then $\mathbf{f}$ can be recovered from $\mathbf{y}$ with an overwhelming probability. Therefore, we need a basis[1] Ψ and a set of sensing vectors with isotropy property such that $\mathbf{f}$ is compressible under the Ψ-transform and the columns of the transformation matrix $\mathbf{\Psi}$ have the least coherence with the sensing vectors. In this chapter, we propose a novel network coding technique that fulfills these conditions.

7.1.2 Signal recovery

In order to recover $\mathbf{f}$ from $\mathbf{y}$, first we need to solve the following convex optimization problem Candes and Plan [2011].

$$\underset{\tilde{\mathbf{x}}\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{2}\|\mathbf{\Phi}\mathbf{\Psi}\tilde{\mathbf{x}} - \mathbf{y}\|_2^2 \ + \ \lambda\sigma\|\tilde{\mathbf{x}}\|_1 \tag{7.5}$$

where $\lambda = 10\sqrt{m \log n}$, $\|\cdot\|_1$ is the norm-1 operator and $\|\cdot\|_2$ is the norm-2 operator.

Some of the efficient and accurate algorithms for solving this problem can be found in Kim et al. [2007] and Blumensath and Davies [2009]. If $\hat{\mathbf{x}}$ is the solution to the convex optimization problem in Equation 7.5, then $\hat{\mathbf{f}} = \mathbf{\Psi}\hat{\mathbf{x}}$ will estimate the original signal $\mathbf{f}$ with an error bounded by $\text{polylog}(n)(s/m)\sigma^2$ Candes and Plan [2011]. In practice, signal $\mathbf{f}$ is not strictly sparse under the Ψ-transform. Instead, $\mathbf{x}$ has a few components with larger magnitudes and most of its remaining components are nearly zero. Suppose that $\mathbf{x}_s$ is a sparse approximation of $\mathbf{x}$ by keeping s most significant components of $\mathbf{x}$ and zeroing its remaining $(n - s)$ components. It is shown that, in such a case the recovery error will not grow much more than $O(\|\mathbf{x} - \mathbf{x}_s\|_2)$ Candes and Plan [2011]; Candes [2006].

[1] A basis for $\mathbb{R}^n$ is a set of vectors $\boldsymbol{\psi}_i, i \in \{1, \ldots, n\}$, such that any vector $\mathbf{f} \in \mathbb{R}^n$ can be represented as $\mathbf{f} = \sum_{i=1}^{n} x_i \boldsymbol{\psi}_i$ where x_i are called the coefficients of $\mathbf{f}$ in basis Ψ. The corresponding transformation matrix $\mathbf{\Psi}$ is made by putting the vectors $\boldsymbol{\psi}_i$ in a matrix, i.e., $\mathbf{\Psi} = [\boldsymbol{\psi}_1 \ \boldsymbol{\psi}_2 \ \ldots \ \boldsymbol{\psi}_n]$ and $\mathbf{f} = \mathbf{\Psi}\mathbf{x}$ where $\mathbf{x} = [x_1 \ x_2 \ \ldots \ x_n]^T$.

7.2 Distributed compression and predistribution via randomized gossiping

Randomized gossiping is a data dissemination approach which employs average consensus algorithm Rabbat et al. [2006]. It makes use of gossiping average consensus technique to compute and distribute random projections of the sensed data. These random projections are the measurements y_j where $j \in \{1, 2, ..., m\}$.

Let $\boldsymbol{\alpha}_i$ refer to the ith column of measurement matrix $\boldsymbol{\Phi}$. Note that $\boldsymbol{\alpha}_i$ is generated in a decentralized manner by employing a pseudo-random number generator. Suppose that SN i computes $f_i\boldsymbol{\alpha}_i$ and stores the result in an array of m real numbers. Let $\mathbf{w}_i[t] \in \mathbb{R}^m$ refer to the content of the array inside SN i at time t. SN l_1 is activated uniformly at random at time t and chooses one of its neighbors l_2 uniformly at random. SN l_1 and SN l_2 exchange $\mathbf{w}_{l_1}[t]$ and $\mathbf{w}_{l_2}[t]$ and update $\mathbf{w}_{l_1}[t+1] = \mathbf{w}_{l_2}[t+1] = (\mathbf{w}_{l_1}[t] + \mathbf{w}_{l_2}[t])/2$.

According to the work by Rabbat, when $t \to \infty$, $\mathbf{w}_i \to \boldsymbol{\Phi}\mathbf{f}$ for all $i \in \{1, 2, \ldots, n\}$ Rabbat et al. [2006]. Therefore, after *sufficiently many* iterations of this protocol, the content of array $\mathbf{w}_i$ in all SNs will *get close enough* to $\mathbf{y} = \boldsymbol{\Phi}\mathbf{f}$. According to the discussion in Section 7.1.2, signal $\mathbf{f}$ can be recovered at any SN after solving Equation 7.5. Since the size of $\mathbf{y}$ and also the array $\mathbf{w}_i$ is in the order of $\mu s \log n$, randomized gossiping requires $O(\mu s \log n)$ transmissions per iteration. The number of required iterations depends on the network topology Rabbat et al. [2006].

If we assume that $s \log n$ measurements are required for accurate signal recovery, then $2s \log n$ transmissions per iteration is performed by the two SNs that are randomly chosen for data exchange. One drawback of this method is that the measurements are inaccurate unless sufficiently many iterations of this protocol are executed. In practice, too many iterations and message exchanges are required to achieve the consensus below an acceptable error threshold.

7.2.1 Comprensus and randomized gossipting

In practice, achieving a negligible recovery error requires a large number of iterations. This is one important drawback of randomized gossiping. Comprensus proves to be more efficient in terms of number of iterations and transmissions required for data dissemination. The novelty of the Comprensus is its efficient network coding technique, which makes a faster convergence to the minimum recovery error possible. In Comprensus, each SN receives a different measurement vector y_i where $i \in \{1, 2, ..., n\}$. For all of these

measurement vectors, some necessary conditions are satisfied ensuring that accurate signal recovery is possible.

We propose a novel network coding mechanism which is still as simple as consensus, nevertheless, requires less time and communications to disseminate the measurements. In randomized gossiping, the SNs run a protocol such that all of them *converge* to a measurement vector $\mathbf{y}$ which is common among all SNs Rabbat et al. [2006].

In Comprensus, each SN receives a different measurement vector $\mathbf{y}_i$, $i \in \{1, 2, \ldots, n\}$. We show that for all of these measurement vectors, the isotropy and incoherence properties hold. Therefore, Comprensus does not need too many iterations for convergence to the same measurement vector among all SNs. Instead, our proposed method guarantees that the same signal $\mathbf{f}$ is accurately recoverable from each individual measurement vector $\mathbf{y}_i$ received by SN i, $i \in \{1, 2, \ldots, n\}$.

After calculating, disseminating and acquiring the measurements, the original signal can be recovered by solving a convex optimization problem as explained in Section 3.2.3, while minimizing the expected squared error.

7.3 The Comprensus protocol

In this section, we explain Comprensus, a simple distributed protocol to disseminate random linear measurements in a WSN with static topology. We assume that the network topology corresponds to a connected regular graph of degree d. It is easy to create a regular graph topology in a WSN when $n \times d$ is an even number. For a given degree d each SN selects at most d neighbors with the highest Received Signal Strength Indicator (RSSI) assuming that each SN has at least d SNs in its communication range Santi [2005]. At the end of this process, we will have a topology corresponding to a regular graph of degree d.

We start by defining the variables and definitions used in our protocol. Suppose that each SN is equipped with two pseudo-random number generators as defined below.

- *Rademacher random generator* produces either $+1$ or -1 each with probability $1/2$.
- *Bernoulli random generator* produces 1 with probability $p = k/n$ and 0 with probability $1 - p$.

We assume that SN i keeps a real number u_i in its internal memory. SN i also keeps a list L_i of real numbers in its memory. The data type of

the elements of L_i is the same as the data type of u_i. Memory requirement for this list is $O(\mu s \log n)$ items. We will see shortly that μ will be a small constant. This list actually holds the random linear measurements which are used thereafter for signal recovery. One can have an estimation of s in an appropriate basis Ψ based on a previous knowledge about the data gathered from the WSN. Since this estimation is not necessarily accurate, it is recommended to use a worst case estimation for s in a real-world deployment of Comprensus.

7.3.1 Distributed Comprensus algorithm

The Comprensus protocol is executed in three phases: *Initialization*, *Dissemination* and *Recovery*. The instructions described below will be executed in parallel by every SN i, $i \in \{1, 2, \ldots, n\}$.

Initialization

First, the list L_i is emptied. Then, SN i reads the value f_i from its sensor and stores it into variable u_i. We assume that each SN is given a unique id and initializes the seeds of the Rademacher and Bernoulli random generators by its id. By choosing an efficient and reliable pseudo-random number generator we minimize the chance that two SNs generate the same sequence of random values Niederreiter [1991].

Dissemination

This phase is repeated r times in parallel by all n SNs. At each iteration $t \in \{1, 2, \ldots, r\}$ all of the SNs execute Algorithm 2 simultaneously.

- $h_i[t]$ is the value generated by the Rademacher random generator of SN i at iteration t.

- $b_i[t]$ is the value generated by the Bernoulli random generator of SN i at iteration t.

1: $u_i \leftarrow h_i[t] \cdot u_i$
2: **if** $b_i[t] = 1$ **then**
3: Transmit u_i
4: **else**
5: **for all** SN j in neighborhood of SN i **do**
6: **if** SN j is transmitting the value $u_j[t]$ **then**
7: $u_i \leftarrow u_i + u_j[t]/n$
8: **end if**
9: **end for**
10: **end if**
11: **if** at least one neighbor has transmitted **then**
12: add u_i to the rear of L_i
13: **end if**

Algorithm 2: Dissemination phase of Comprensus

Remark: We assume that the SNs cannot transmit and listen at the same time. Also the wireless channel of two adjacent nodes cannot be used simultaneously. Therefore, if a set of adjacent SNs want to transmit at the same iteration, they transmit one by one according to the descending order of their ids. They aggregate their received measurements in a temporary variable and update their corresponding u_i only after all of these concurrent transmissions are completed.

If the SNs are perfectly synchronized, aggregation by signal superposition helps to perform the dissemination phase faster. Signal superposition allows multiple nodes to transmit simultaneously and the receiver accumulates the received values at the same time Bajwa et al. [2007, 2006].

Recovery

SN i derives a vector $\mathbf{y}_i$ by stacking the entries in list L_i. When all SNs agree on a common random generator algorithm, the linear combinations that led to the values in L_i are reproducible as described in Section 3.3. These linear measurements are then placed in Equation 7.5 to recover $\mathbf{f}$. We show in Section 7.3.3 that the linear measurements acquired in the dissemination phase obey the isotropy condition and have low coherence with DCT.

Line 1 of Algorithm 2 generates a new Rademacher value and multiplies it by the current value of u_i which is first set to f_i in the initialization phase. Line 2 decides whether SN i is to transmit in this iteration or not. Since $b_i[t]$ returns 1 with probability k/n, this is equivalent to the case that almost k out of n SNs select themselves uniformly at random to transmit. Executing the line 3 consumes the most amount of battery power, as using the radio in transmitting mode is the major energy drain of a SN Mahfoudh and Minet

[2008]. If SN i is not in transmitting mode at iteration t, i.e., $b_i[t] = 0$, then it listens to the communication channel and accumulates the values sent by neighboring nodes onto u_i after dividing them by n as instructed in lines 5 through 9. Summing the received values from neighboring nodes can be done arithmetically by using a simple Time Division Multiple Access (TDMA) mechanism Akyildiz et al. [2002]. A faster alternative is signal superposition when the SNs are perfectly synchronized Bajwa et al. [2007, 2006]. It can also happen that no neighbor of SN i does a transmission at iteration t. In this case, no value is added to the list L_i. This condition is checked in line 11, and thus, line 12 is executed only when at least one neighboring node has transmitted. We will explain shortly why this restriction is necessary.

7.3.2 Matrix representation of the distributed protocol

In this section we examine the network-wide implication of Algorithm 2 by using the equivalent matrix representation of Comprensus.

Let N_i denote the set of the d neighbors of SN i.

Definition 17. *Transition matrix* $\mathbf{M}_t$ *at iteration* t *is an* $n \times n$ *real matrix with the following attributes.*

1) $M_t[i,i] = h_i[t]$ *for* $1 \leq i \leq n$.

2) $M_t[j,i] = h_i[t]/n$ *when* $j \in N_i$ *and* $b_i[t] = 1$.

It is easy to verify that after iteration t of the dissemination phase,

$$\begin{pmatrix} u_1[t] \\ u_2[t] \\ \vdots \\ u_n[t] \end{pmatrix} = (\mathbf{M}_t \times \mathbf{M}_{t-1} \times \cdots \times \mathbf{M}_1)\,\mathbf{f} \tag{7.6}$$

describes the contents of variables u_i , $i \in \{1, 2, \ldots, n\}$. We address the effect of noise at the end of our matrix analysis. We also define the $n \times n$ matrix $\mathbf{Q}_t$ as

$$\mathbf{Q}_t := \begin{pmatrix} \mathbf{q}_{1,t} \\ \mathbf{q}_{2,t} \\ \vdots \\ \mathbf{q}_{n,t} \end{pmatrix} := \mathbf{M}_t \times \mathbf{M}_{t-1} \times \cdots \times \mathbf{M}_1 \tag{7.7}$$

where $\mathbf{q}_{1,t}$, $\mathbf{q}_{2,t}$, ... , $\mathbf{q}_{n,t}$ are the rows of matrix $\mathbf{Q}_t$.

We define a set R_i as

$$R_i := \{t \quad | \quad \exists j \in N_i \ : b_j[t] = 1\} \tag{7.8}$$

to refer to the set of iterations in which at least one neighboring node of SN i is transmitting. We also define matrix $\boldsymbol{\mathcal{A}}_i$ as

$$\boldsymbol{\mathcal{A}}_i := [\mathbf{q}_{i,t_1}^T \ \mathbf{q}_{i,t_2}^T \ \dots \ \mathbf{q}_{i,t_{m(i)}}^T]^T \tag{7.9}$$

where $m(i) = |R_i|$ is the number of measurements received by SN i and $\{t_1, t_2, \dots, t_{m(i)}\} = R_i$. The number of received measurements may differ from one SN to other. Nevertheless, when the network topology corresponds to a regular graph, all of the nodes are expected to receive almost the same amount of measurements, since each SN has an equal chance to transmit and receive messages. It can be shown that the measurement vector $\mathbf{y}^{(i)}$ made by stacking the values in list L_i will be

$$\mathbf{y}^{(i)} = \boldsymbol{\mathcal{A}}_i \mathbf{f} + \mathbf{z} \tag{7.10}$$

where $\mathbf{z}$ is the additive noise. The noise is added either by the communication channel or can be regarded as a side effect of low precision floating pointing storage and processing inside the SNs. We model $\mathbf{z}$ by a white Gaussian noise vector in our simulations and experiments.

If the rows of $\boldsymbol{\mathcal{A}}_i$ obey the isotropy property and have low coherence with a compressive basis, then $\mathbf{f}$ can be recovered at SN i from $\mathbf{y}^{(i)}$ with high probability as detailed in Section 7.1.2. Now the reason for the restriction in Line 11 of Algorithm 2 becomes clear. We only let *newly received* measurements to be aggregated and added to the measurement list L_i. Otherwise, $\boldsymbol{\mathcal{A}}_i$ will have at least two rows which are linearly dependent, and thus, $\boldsymbol{\mathcal{A}}_i$ is not full rank. In other words, we will have redundant measurements stored in L_i if we do not check the condition in Line 11 of Algorithm 2.

Suppose that $\bar{m}$ is the average number of measurements received per SN. $\bar{m}$ should be in order of $O(\mu s \log n)$ to allow successful recovery. When these conditions are fulfilled, the signal vector $\mathbf{f}$ can be recovered at every SN after running the Comprensus protocol. Next, we examine isotropy and incoherence properties of our measurement matrix $\boldsymbol{\mathcal{A}}_i$ for $i \in \{1, 2, \dots, n\}$.

7.3.3 Numerical experiments

In this section, we investigate the isotropy and incoherence of our measurement method through numerical experiments on simulated WSNs. We perform comprehensive numerical experiments on simulated WSNs consisting of $n = 128$ SNs. The network graph is a random regular graph of degree $d = 5$ which is freshly generated in each experiment and the results are averaged over multiple simulation runs.

We let the SNs to generate their corresponding h_i and b_i random numbers and execute the Comprensus protocol for varying values of r and k. Each experiment is run several times and all of the results are averaged to eliminate randomness effects.

In Section 7.1.1, we have seen that even if the set of sensing vectors have the *near-isotropy* property, the signal $\mathbf{f}$ can be recovered from measurement vector $\mathbf{y}$. In Comprensus, the set of the sensing vectors for SN i are the rows of $\boldsymbol{\mathcal{A}}_i$ and the measurement vector for SN i is $\mathbf{y}^{(i)}$. We define a metric for *deviation from isotropy* and show that the rows of $\boldsymbol{\mathcal{A}}_i$ have a very low deviation from the isotropy property.

Definition 18. Deviation from isotropy *for a random sensing vector* $\mathbf{a}$ *is defined as* $\sum_{e\in\mathcal{E}_a}(1-e)^2$ *where* $\mathcal{E}_a$ *is the the set of eigenvalues of the square matrix* $E[\mathbf{a}\mathbf{a}^T]$.

This metric determines how much $E[\mathbf{a}\mathbf{a}^T]$ behaves like an identity matrix. In the ideal case, $E[\mathbf{a}\mathbf{a}^T] = \mathbf{I}$ and has only one eigenvalue, i.e., 1, and thus, the deviation from isotropy is zero. For random sensing vector $\mathbf{a}$, if the eigenvalues of $E[\mathbf{a}\mathbf{a}^T]$ are all very close to 1, $E[\mathbf{a}\mathbf{a}^T]$ behaves like an identity matrix and deviation from isotropy as defined in Definition 18 will be low.

In our numerical experiments, a large set of measurement matrices $\boldsymbol{\mathcal{A}}_i$ are generated. Our random sensing vectors are actually the rows of the randomly generated measurement matrices $\boldsymbol{\mathcal{A}}_i$. For each row $\mathbf{a}^T$ of these matrices, we calculate $\mathbf{a}\mathbf{a}^T$ and sum up all of the results. $E[\mathbf{a}\mathbf{a}^T]$ is then numerically calculated by dividing this summation by the total number of the randomly generated sensing vectors.

Observation 1 – near-isotropy: Deviation from isotropy according to Definition 18 is calculated over all randomly generated measurement matrices $\boldsymbol{\mathcal{A}}_i$, $i \in \{1, 2, \ldots, n\}$ and a full range of experiments with k varying from 1 to n and $r \in \{32, 64, 128\}$. The results as illustrated in Figure 7.1.a prove that our measurement mechanism obeys the near-isotropy property with negligible deviation, i.e., $E[\mathbf{a}\mathbf{a}^T] \approx \mathbf{I}$.

Observation 2 – low coherence: We set $\boldsymbol{\Psi}$ to be the inverse DCT matrix and calculate the coherence with the DCT basis according to Definition 16 for randomly generated measurement matrices with k varying from 1 to n and $r \in \{32, 64, 128\}$. The averaged results over a large set simulations as illustrated in Figure 7.1.b shows that the coherence factor is also low.

Next, we evaluate the performance of Comprensus and compare it with randomized gossiping and an oracle-based approach.

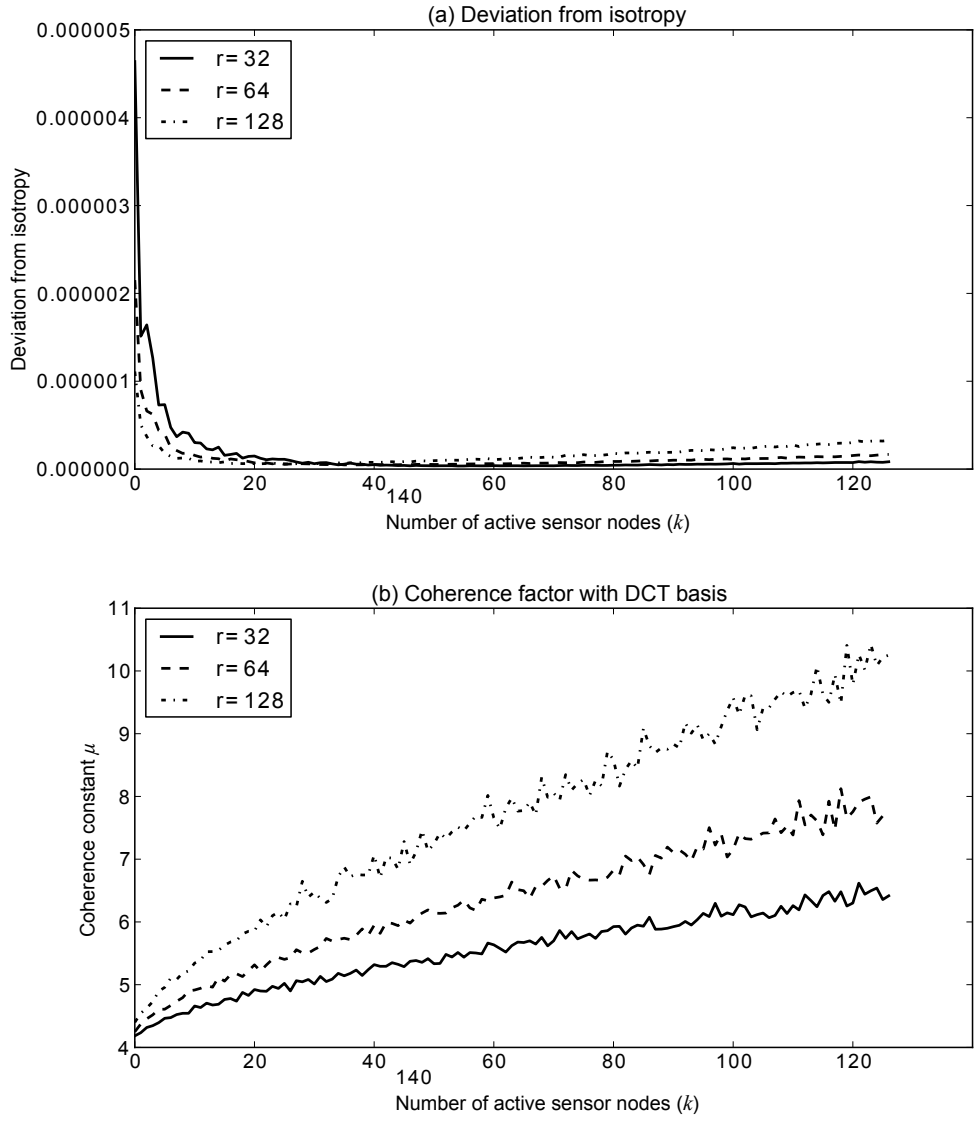


Figure 7.1: Near-isotropy and low coherence of Comprensus measurement scheme

7.4 Evaluation

We simulate the Comprensus protocol by distributed execution of Algorithm 2 for different values of r and k on a large set of synthesized spatial signals. After running each instance of the Comprensus protocol, the Signal to Noise Ratio (SNR) of the recovered signal is calculated. The SNR is measured in decibels (dB) and the recovery algorithm is run at a randomly chosen SN. This guarantees that the recovery is possible at any arbitrarily chosen node.

If $\mathbf{f}$ is the original signal and $\hat{\mathbf{f}}$ is the recovered signal, we define SNR as

$$\text{SNR} := 10 \log_{10} \left(\frac{\|\mathbf{f}\|_2^2}{\|\mathbf{f} - \hat{\mathbf{f}}\|_2^2} \right). \tag{7.11}$$

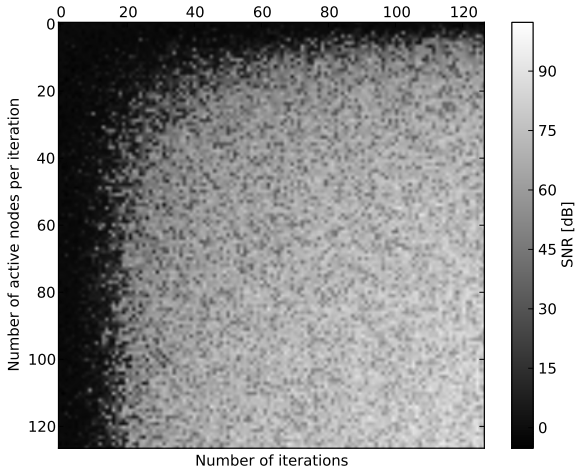
Figure 7.2 shows the results of simulating a network consisting of 128 nodes. Number of iterations and number of active nodes are varying from 1 to 128. Each point on the SNR diagrams corresponds to one simulation run. Brighter area shows higher SNR which means accurate signal recovery at an arbitrary SN and darker area shows lower SNR which means that signal recovery is not possible, and hence, the sensed data is not accessible from an arbitrary node.

An interesting observation in our evaluations is that the transition to the condition where accurate recovery is possible is relatively sharp. Looking at the SNR diagrams of Figure 7.2, the border between the bright area (successful dissemination) and dark area (non-recoverability) has a recognizable contrast. Sharp transition between recoverability and non-recoverability states in CS is comprehensively studied in the Donoho-Tanner universal phase transition inspections Donoho and Tanner [2009].

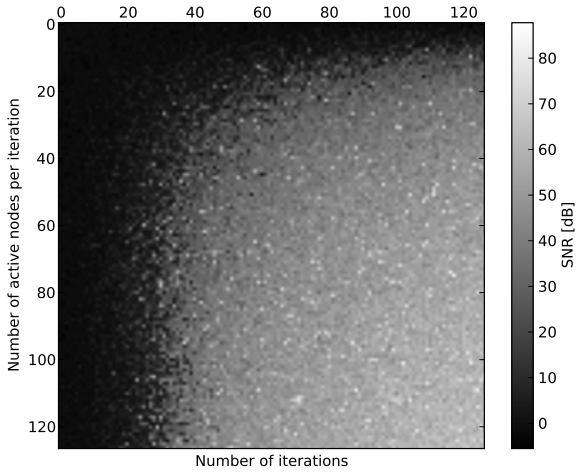
We conduct a large set of simulations with different configurations of the Comprensus protocol and for different values of the sparsity parameter s. Next, we compare Comprensus to randomized gossiping for three illustrative values for s. Our evaluations show that Comprensus outperforms randomized gossiping in terms of message cost and the dissemination time. In Section 7.4.2, we show that Comprensus performs close to the optimal case when no packet loss occurs.

7.4.1 Comparison to randomized gossiping methods

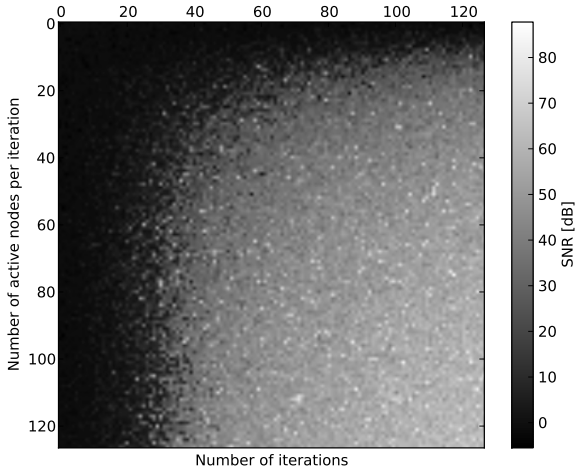
In this section, we compare Comprensus to decentralized compression based on randomized gossiping Rabbat et al. [2006]. We set our SNR requirement for both protocols to 40 dB and compare their performance in achieving this



(a) Signal recovery for $s = 5$



(b) Signal recovery for $s = 10$



(c) Signal recovery for $s = 20$

Figure 7.2: Accuracy of signal recovery for different sparsity levels

requirement. Network topology and all other conditions are also the same for both protocols.

As described in Section 7.2, randomized gossiping requires at least $s \log n$ data exchanges per iteration. The number of required iterations is dependent on the network topology and maximum allowed measurement error. Assume that $\mathbf{w}_i[t]$ is the content of the measurement vector of SN i at iteration t of randomized gossiping as defined in Section 7.2. As we have seen in Section 7.2, the difference between $\mathbf{w}_i[t]$ and $\mathbf{y} = \mathbf{\Phi f}$ shrinks to zero when $t \to \infty$.

We define the measurement error of SN i at iteration t as

$$\epsilon_{i,t} \ := \ \|\mathbf{w}_i[t] - \mathbf{y}\|_2. \tag{7.12}$$

We also define average measurement error at iteration t as $\epsilon_t := (\sum_{i=1}^{n} \epsilon_{i,t})/n$. From the arguments in Section 7.1.2 we know that for recovering $\mathbf{f}$ with SNR of at least 40 dB, the measurement error $\epsilon_{i,t}$ must be lower than $\|\mathbf{f}\|_2 \times 10^{-4}$ for the measurement vector received by SN i at time t.

Figure 7.3 shows how ϵ_t decays with the number of randomized gossiping iterations for a WSN consisting of $n = 128$ SNs with a network topology corresponding to a connected regular graph of degree $d = 5$. We observe that, average measurement error goes below our required threshold after almost 1200 iterations. We round down this number to 1000 iterations in favor of the randomized gossiping method.

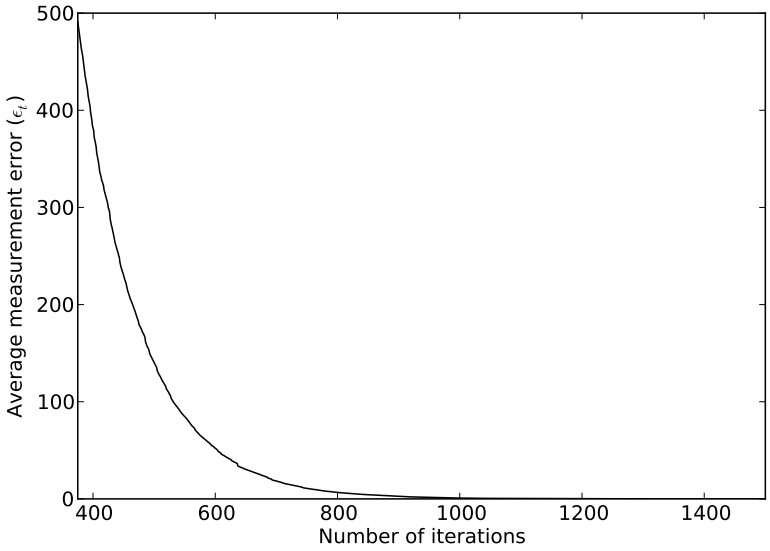


Figure 7.3: Measurement error decay with iterations of randomized gossiping

Now we compare the total amount of transmissions in Comprensus and randomized gossiping. We consider the three test cases illustrated in Figure

7.2. For $s \in \{5, 10, 20\}$, randomized gossiping requires $s \log n \approx 2.1 \times s$ transmissions per iteration when $n = 128$, and thus, almost $2.1 \times 10^3 \times s$ transmissions in total, since it requires to execute 1000 iterations. Comprensus needs rk transmission corresponding to point (r, k) that falls on the bright part of the SNR diagram of Figure 7.2. For s being 5, 10 and 20 we set $(r = 50, k = 30)$, $(r = 60, k = 40)$ and $(r = 80, k = 40)$ respectively. Looking at Figure 7.2, we see that these are rather conservative selections and signal recovery is possible with fewer numbers of transmissions. Nevertheless, we run Comprensus with these conservative settings and compare its performance to the randomized gossiping method. The comparison result is summarized in Table 7.1.

Table 7.1: Comparing Comprensus to randomized gossiping

s	Total number of transmissions	
	Comprensus	Randomized gossiping
5	1.5×10^3	1.0×10^4
10	2.4×10^3	2.1×10^4
20	3.2×10^3	4.2×10^4

Comprensus proves to disseminate the random linear measurements not only in significantly less number of iterations, but also using much less amount of in-network transmissions. Using its efficient network coding technique, Comprensus disseminates compressible data with low latency and high quality while keeping the number of transmissions as low as possible in order to preserve more battery power of the SNs.

7.4.2 Comparison to oracle-based approach

Comparison with an oracle-based approach gives us a better understanding of Comprensus' performance in comparison to the optimal solution. We assume that an oracle knows all of the sensed data, i.e., the vector $\mathbf{f}$ in advance, and hence, it also knows its sparse transform, i.e., $\mathbf{x}$. The oracle broadcasts only the s significant coefficients of $\mathbf{x}$ into the network. Thus, the communication cost of the oracle-based approach is $O(sn)$. Note that broadcasting is performed hop by hop. For a limited number of neighboring hops, the broadcast of a single data item requires $O(n)$ transmissions, and

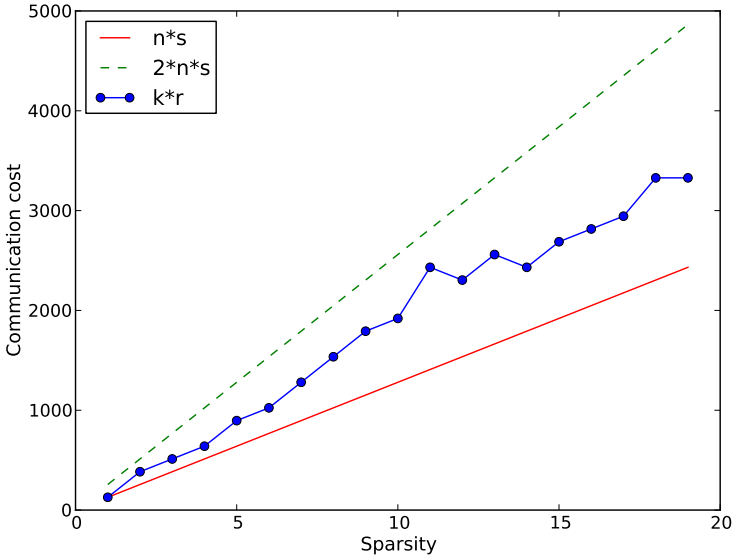


Figure 7.4: Communication cost for SNR-threshold of 30 dB

thus, the total number of transmissions for broadcasting the s significant coefficients of $\mathbf{x}$ is $O(sn)$.

Figure 7.4, Figure 7.5, Figure 7.6, demonstrate communication cost of Comprensus for different values of sparsity and for 30 dB 35 dB and 40 dB SNR-thresholds respectively. We observe that, the communication cost of Comprensus is bounded between $O(s \times n)$ and $O(3 \times s \times n)$. These numerical experiments indicate that Comprensus functions almost optimally under the conditions that are applied for the simulation. A generalization of these results or a formal proof of optimality of Comprensus is regarded as an interesting direction for future work.

In Appendix A we include more diagrams extracted from the database of our simulation results.

7.5 Chapter summary

In this chapter, we studied dissemination of compressible data using local information exchanges between the nodes in a wireless sensor network. Our approach is based on the theory of compressed sensing. We present a novel network coding protocol, named Comprensus, that enables each sensor node of a wireless sensor network to operate potentially as a sink. Our agile sink

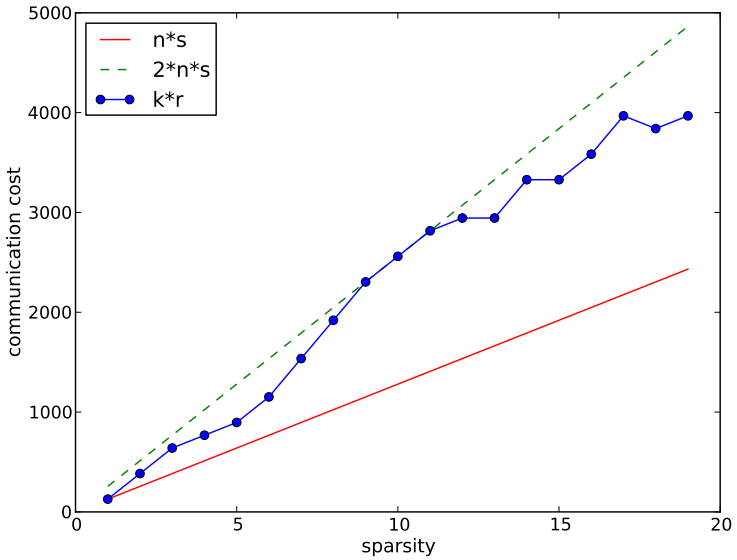


Figure 7.5: Communication cost for SNR-threshold of 35 dB

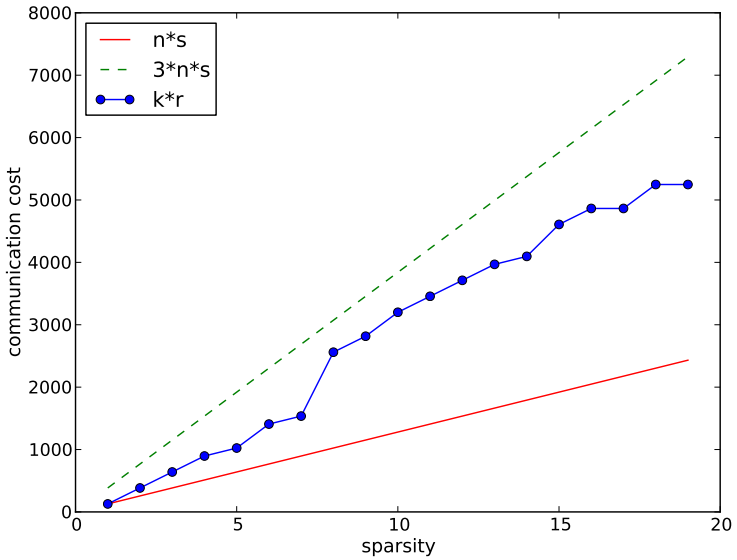


Figure 7.6: Communication cost for SNR-threshold of 40 dB

selection techniques can avail the full set of the sensed data by querying a small amount of measurements from *any* arbitrary node in the network. The techniques proposed in this chapter are particulary suitable for scenarios where the sink of the wireless sensor network is mobile or when each node should access the global state of the environment.

The main advantage of the Comprensus protocol is its simple dissemination algorithm that is easily implementable on the scarce hardware resources of the sensor nodes. The complex part of the protocol, i.e., signal recovery is offloaded to an external sink or data collector that possess enough computation power.

Our evaluations show that Comprensus outperforms the state of the art methods for data dissemination in wireless sensor networks that are based on compressed sensing both in terms of communication cost and the time required for data dissemination. Our approach benefits from inherent resilience of compressed sensing to communication and measurement noise. Moreover, comprehensive simulation and experiments of our method show a nearly optimal performance of the Comprensus protocol in a noiseless scenario. Our experiments provide a basis for further theoretical and practical investigations of our proposed method, especially for dynamic topologies.

Chapter 8

Conclusions and Future Research

The main distinguishing aspect of WSNs from other sensory systems, is their ability to encode the data while it is being transmitted to the base station. Therefore, distributed network coding techniques have been always in focus of designing effective data collection techniques for WSNs. We have seen that in general there is a trade-off between decentralization of encoding and the efficiency and or overhead of the data collection technique. Less centralized processing often requires less coordination overhead. On the other hand, centralized processing of data helps data compression or aggregation methods to achieve a better compression ratio.

The trade-off between compression ratio and centralized processing overhead, makes in-network compression particularly challenging for WSNs. Especially, because the limited computation and energy resources of the SNs does not allow complex computation and high-rate data transfer.

Compressed Sensing (CS) is a sub-sampling method that allows a good integration of in-network compression in WSNs. We have seen that CS provides a well-defined and quantitative trade-off measurement between the cost of data collection and quality of information that is received by the sink.

The CS-based methods discussed in this thesis provide a multi-tier solution that covers different layers of WSN sensory systems, from sampling up to data transfer and information reconstruction at the sink. At each of the data collection steps, some unwanted distortion may be present that decreases the quality of the information received by the sink. While CS intrinsically provides a certain level of robustness, we have extended the existing techniques to better handle failure models of WSNs such as node or link failures.

8.1 Contributions of this thesis

This thesis focuses on data reduction techniques in WSNs that are based on the CS theory. CS makes it possible to acquire signals more efficiently in scenarios where collecting all samples of the signal is either infeasible or costly. The contributions of this thesis can be summarized as the following:

8.1.1 Reordering technique for better signal compressibility

We have shown that samples of spatial signals can be reordered to provide a more compressible view of the phenomena that are being recorded by a WSN. In most of WSN deployments, the numbering or indexing of the SNs is done by convention. The identification numbers of the SNs are labels that are given to those SNs and can be changed to produce a new ordering of the samples. In Chapter 4 we discussed techniques to define a mapping between the physical identification indices of the SNs and the sequence of the samples in the permuted signal vector. Also a polynomial-time algorithm to find a more compressible permutation of the samples is provided. The algorithm accepts a set of sensed data and a compressive basis Ψ. Our proposed algorithm finds a permutation of the samples that is more compressible in Ψ. In our simulation, we run the algorithm at a time instance t and the WSN continues its sensing operation to collect the data at time $t + 1$. The SNs are relabeled or re-indexed according to the more compressible permutation calculated from the previous sampling round. After running multiple simulations with different configurations, we conclude that compressible ordering leads to a better performance of compressed sensing at time instance $t + 1$.

8.1.2 Concept of sliding sampling window for spatio-temporal compressed sensing in WSNs

As discussed in Chapter 2, data compression algorithms are mainly based on finding a correlation between a large amount of data. The more structured the data is, the higher compression ratio can be achieved by the compression algorithm. Especially when the structure or correlation is found for a larger set of data, better data reduction is expectable. Larger data sets in time domain requires longer delays to gather the data over an extended time period. A larger sampling window can improve the efficiency of compressed sensing, but requires a longer time to fill the sampling buffer.

We have presented the concept and implementation of a sliding sampling

window in Chapter 5 to solve the issue of longer delays when the sampling period is extended. Our proposed method requires an initialization phase to fill its sampling buffer. After the initialization phase, the data is delivered to the user without any additional delay. Our concept of sliding sampling window takes advantage of the temporal compression as well spatial compression. At the same time, it does not introduce additional delays for sampling the data. The penalty of refilling the sampling buffer occurs only when a non-recoverable error failure happens. In most cases of transient failures, the location of the failure is detected.

8.1.3 Methods to detect and isolate the failing nodes

Node failure and link breakage are two of the main challenges towards an efficient data collection technique for WSNs. In traditional compression techniques, the value of data packages increases drastically when the data is compressed, because the loss of those packages means losing a larger amount of raw data. In compressed sensing, the measurements are all considered to carry an equally valuable amount of data.

In our proposed methods, failures of the sensor nodes are modeled as abnormal events. We propose an *event detection* mechanism for our spatiotemporal sampling window mechanism. Our method uses an over-complete dictionary in its signal reconstruction phase to cancel the negative impact of missing samples caused by the failing nodes or the broken links.

In a second approach that suits better for WSNs with chain or linear topology, we proposed using Haar wavelet domain as the compressive domain to detect the location of the missing samples that are caused by network failures. This is possible when we elevate the values sensed by the SNs by an offset that is much larger than the SNs' valid sensing range. Elevation of the signal allows us to distinguish the locations of the signal with zero value from the locations where the SN actually sensed a value of zero for the physical parameter.

After detecting the location of the node failures, the failing nodes are excluded (isolated) from the data reconstruction process. Only that part of the sensed data that is successfully transmitted is reconstructed. Our evaluations show that the quality of the reconstructed signal is much higher when the erroneous samples are excluded from the signal recovery process.

8.1.4 Compressive signal dissemination in WSNs

Dissemination of the sensed signal can be a very costly task for a WSN, because it usually requires a lot of in-network transmissions. Note that in

signal dissemination, all of the sensed data is to be accessible to all nodes of the network. It must be possible to recover or reconstruct the global state of the environment by fetching data from any node of the network. Most of the existing compression techniques, including the sate of the art methods that are based on compressed sensing are optimized towards gathering the data at a single point, i.e., the sink. Extensions to support multiple sinks are basically generalizations of the central data collection techniques.

This thesis introduces a novel network coding technique based on compressed sensing that is specifically designed for the purpose of signal dissemination in WSNs. It allows dissemination of the sensed data to all sensor nodes without collecting it at a central point. Our coding mechanism which is called *Comprensus* only involves the exchange of numerical values between neighboring nodes and performing simple arithmetic operations, i.e., addition and multiplication. Comprensus disseminates a special linear combination of the sample values. We have shown that it is possible to reconstruct the whole sensed data by applying the recovery algorithm of compressed sensing to the accumulated linear combinations that are stored in any of the sensor nodes.

8.2 Lessons learned

In this thesis, we have have introduced novel data collection techniques by following a new paradigm of distributed data gathering in WSNs. In most of the state-of-the-art works, sampling, encoding, compression and failure handling are regarded as separate problems. This thesis provides a simpler and yet more effective data collection technique by applying compressed sensing as a holistic theory that combines sampling, encoding and compression into a distributed network coding technique. This network coding is easy to implement and does not require too much prior information about the characteristics of the signal that is to be sampled.

The main lesson to learn here is that fundamental rethinking of theories for distributed coding has led to design of better data collection methods. Such novel designs would not be achievable if the problems of sampling, encoding, transmission and failure handling are treated separately and independent from each other. The theory of compressed sensing which is the main focus of this thesis allows designing data gathering techniques that span over different stages of data collection in WSNs, including the sampling stage, encoding, data transmission and network coding, and finally, post-processing and decoding of the collected data. By focusing on the characteristic requirements of WSNs, we have extended the theory of compressed sensing to per-

form saptio-temporal distributed sampling with abnormal event detection in WSNs. Most notably, the methods and techniques that we presented in this thesis do not modify the sampling and network coding parts of compressed sensing. They mainly provide new signal reconstruction methods. Therefore, most of the techniques presented in this thesis can be implemented at the sink of a WSN by having very little or no changes to the hardware or software of the sensor nodes.

8.3 Future work

Our last two contributions that was described in chapters 6 and 7 can open new ways towards application of compressed sensing in WSNs with mobile nodes. This can involve an ad-hoc network of mobile nodes such as robots that require accessing the sensed data from other robots to achieve a specific goal. The main challenge of applying compressed sensing in a network with mobile nodes, is that the topology of the network is not fixed. All of the methods described in this thesis target WSNs with stationary sensor nodes. In Chapter 6 we consider a network with stationary sensor nodes, but the topology of the network can be other than what is fixed during the network initialization. Our robust compressed sensing method allows the network topology to be altered during the run time of the WSN. However, topology changes are captured in form of link and or node failures. The idea of using over-complete dictionaries and further post-processing during signal reconstruction can be extended to support more complex topology changes that occur in a WSN with mobile nodes.

In network of mobile sensor nodes, or mobile sensor and actuator nodes, it may not be necessary or even feasible to designate a specific node a fixed set of nodes as the sink(s) of the WSN. For example, mutual localization aims to localize the robots in a multi-robot configuration relatively to each other Franchi et al. [2009]. The robots are equipped with some sensors, such as ranging sensors. The goal of mutual localization is to calculate the location of each robot relative to its neighboring robots by performing calculations on the perceived data by all of the robots. This requires efficient sharing of the sensed signals. Efficient signal sharing can be achieved by extending the Comprensus protocol discussed in Chapter 7 to mobile topologies.

The application mentioned here is only one example of possible directions to extend the techniques that are developed in this thesis. There are also rooms for improvement of the introduced methodologies. As another example, better event detection techniques can be explored to increase the detection and isolation capability of our robust compressed sensing scheme

introduced in Chapter 6. The compressed sensing theory itself is a an evolving research area. New network coding or signal reconstruction algorithms that are discovered in future can be integrated into the existing frameworks that are presented in this thesis. Our proposed methods allow the flexibility of of choosing other measurement ensembles or reconstruction algorithms other than what used in this thesis. Depending on application requirements, individual modules of our frameworks can be altered or improved to achieve even better performance.

Appendices

Appendix A

Detailed evaluation of Comprensus

This appendix provides more detailed evaluation results of simulating the Comprensus protocol introduced in Chapter 7.

We have seen in Chapter 7 that the compressibility of the spatial signal plays a crucial role in the communication cost of the Comprensus protocol. Higher compressibility of the signal, i.e., smaller s, requires a less number of active nodes per iteration and or less number of iterations overall. The communication cost is the total number of message exchanges which is equal to the multiplication of the iteration counts and active nodes per each iteration.

Figures A.1 to A.19 illustrate further simulation results when the sparsity parameter s is set to $1, 2, 3, \ldots, 19$. Note that the compressibility of the signal decreases as s increases. Therefore, the communication cost increases for larger values of s. The communication cost diagrams presented here include the whole set of simulation results with many different configurations. The network topology in each simulation is generated randomly, and hence, each simulation is different from other simulation instances.

The diagrams presented here give a very good understanding of Comprensus' performance when it is run in many different simulation setups. Each dot in Figures A.1 to A.19 corresponds to one run of Comprensus data dissemination and recovery. We observe from these diagrams, that Comprensus quickly achieves accurate signal recovery, i.e., higher Signal to Noise Ratio (SNR), as the number of exchanged messages starts to increase from the minimum. This means that with slightly more communication cost, it is possible to achieve an accurate dissemination. Usually a very high quality is achievable with not a so much communication cost.

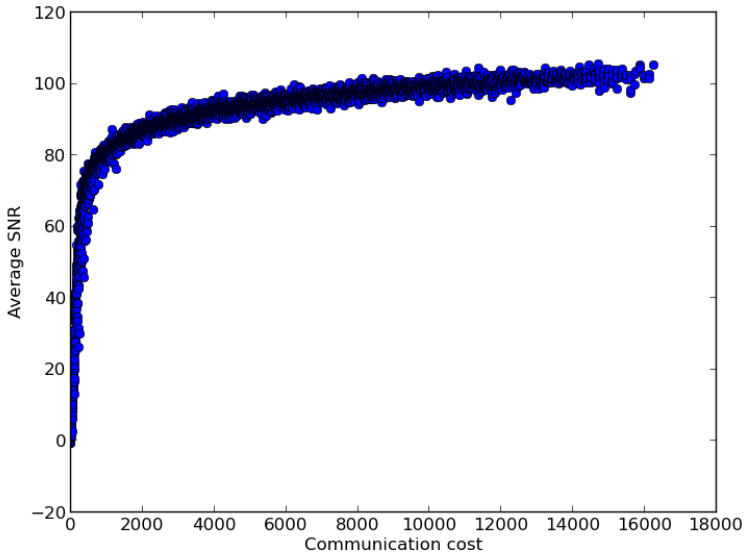


Figure A.1: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 1

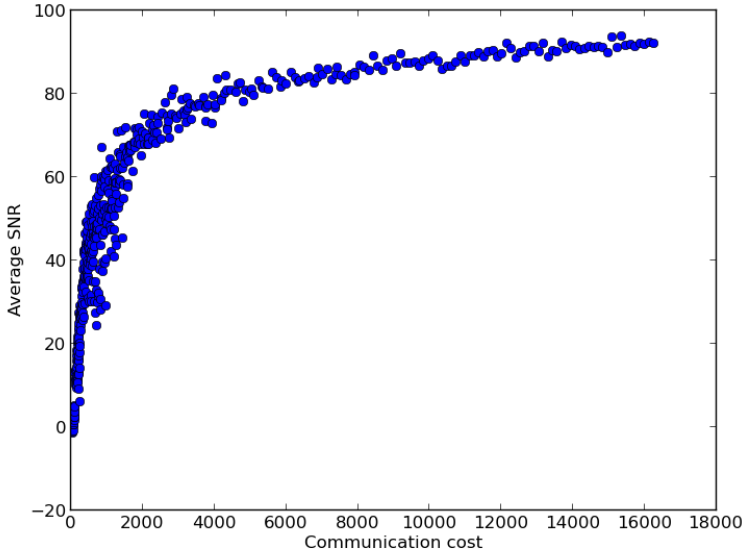


Figure A.2: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 2

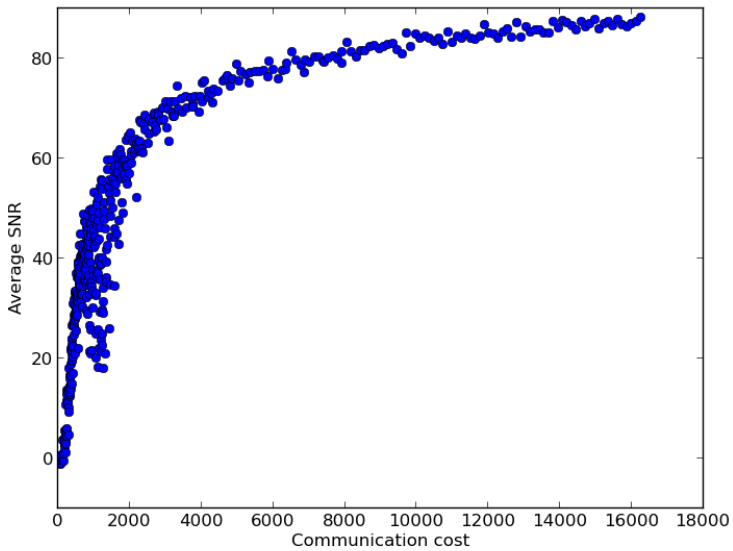


Figure A.3: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 3

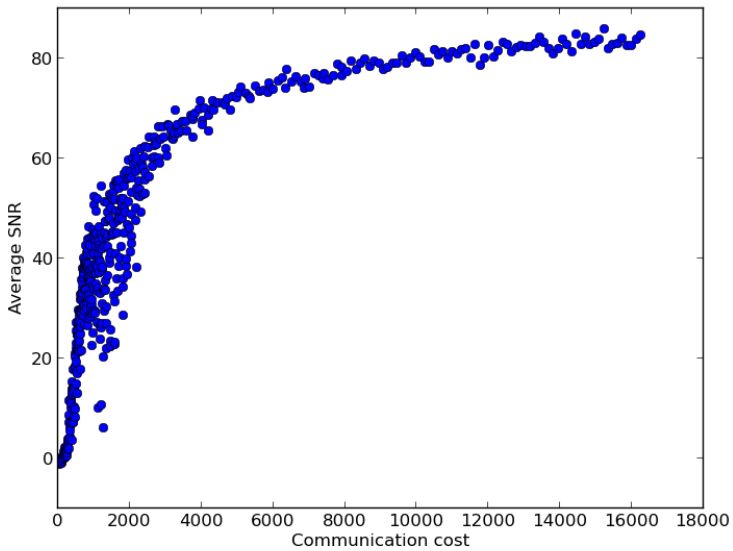


Figure A.4: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 4

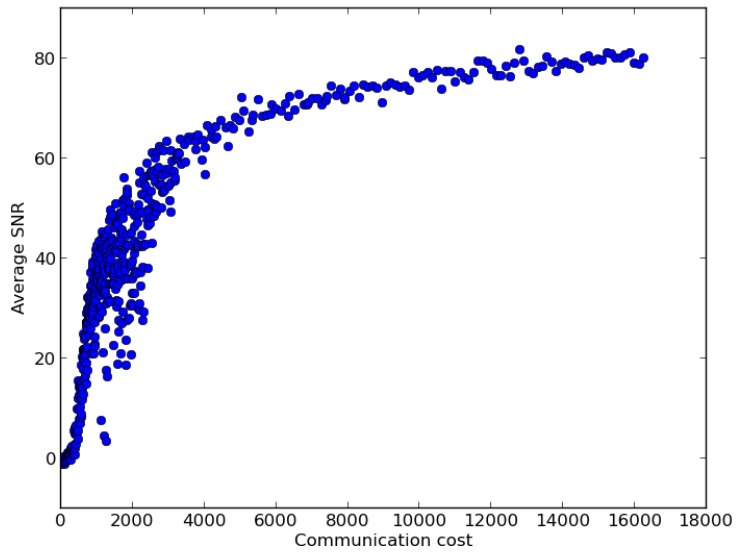


Figure A.5: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 5

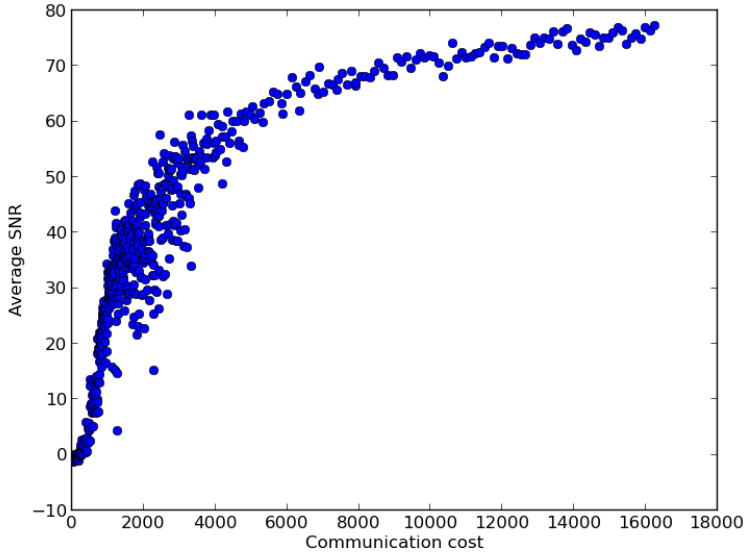


Figure A.6: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 6

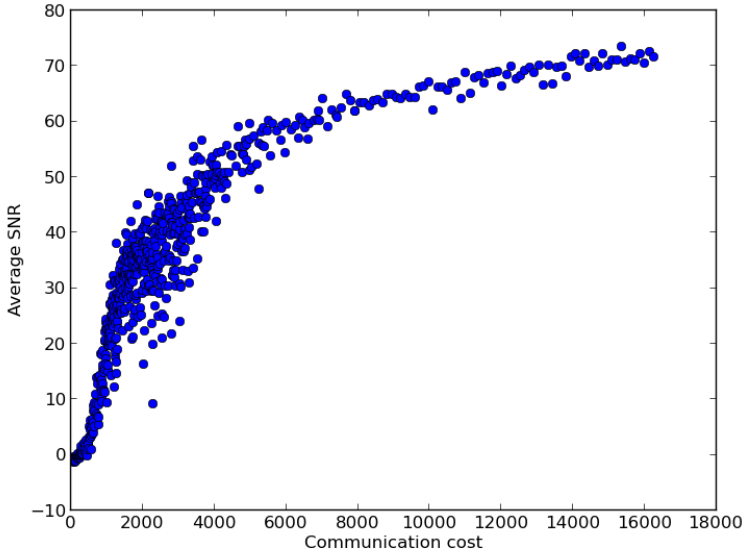


Figure A.7: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 7

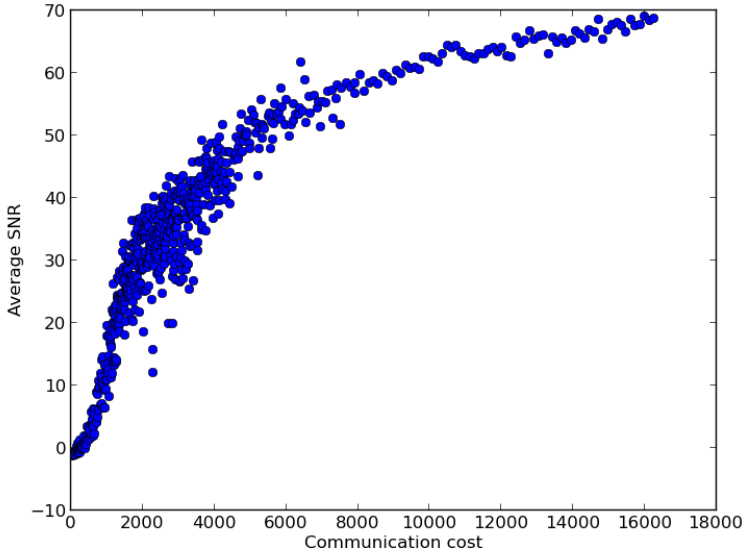


Figure A.8: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 8

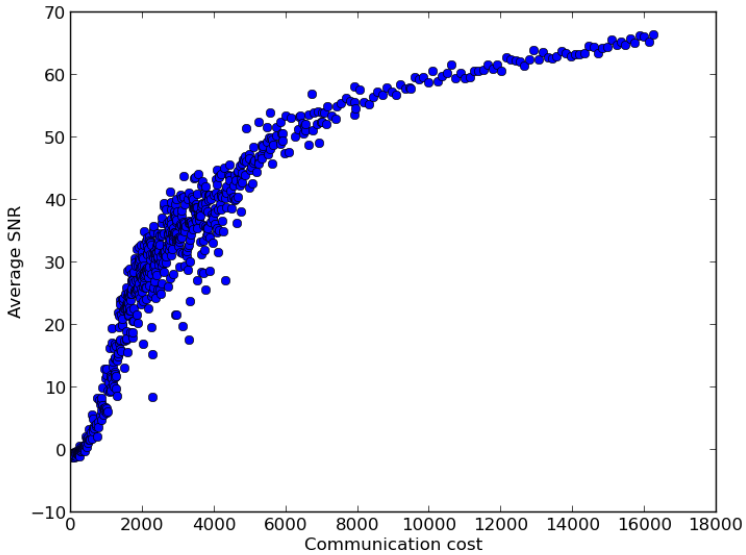


Figure A.9: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 9

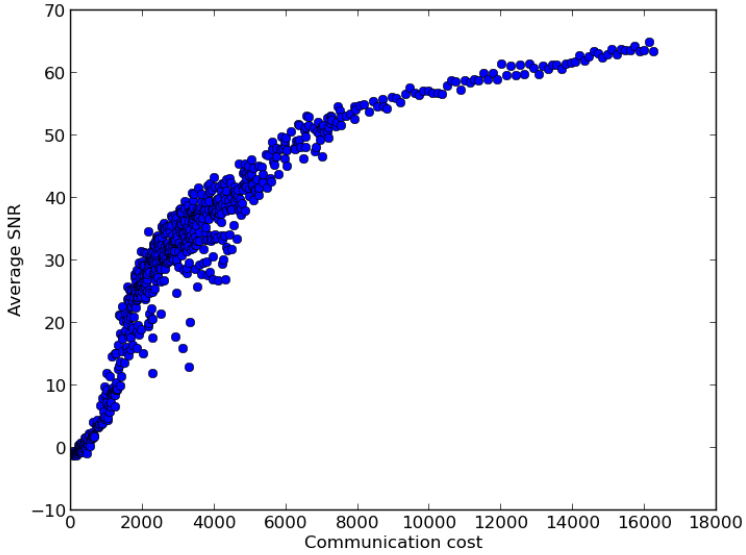


Figure A.10: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 10

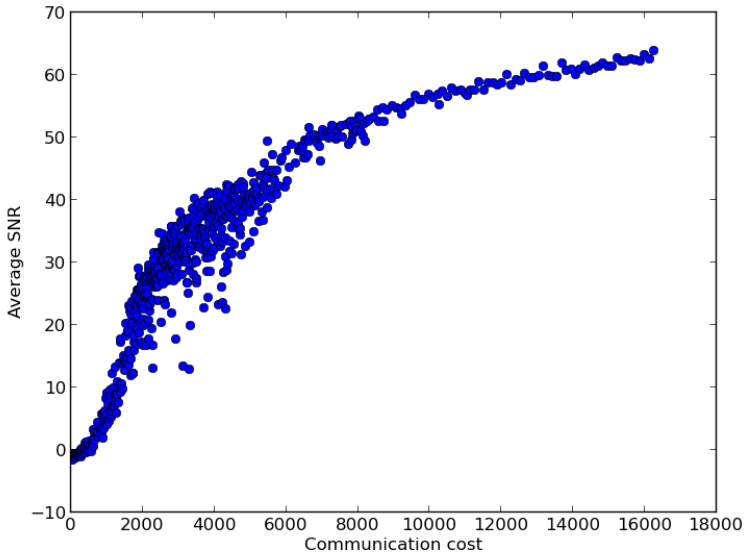


Figure A.11: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 11

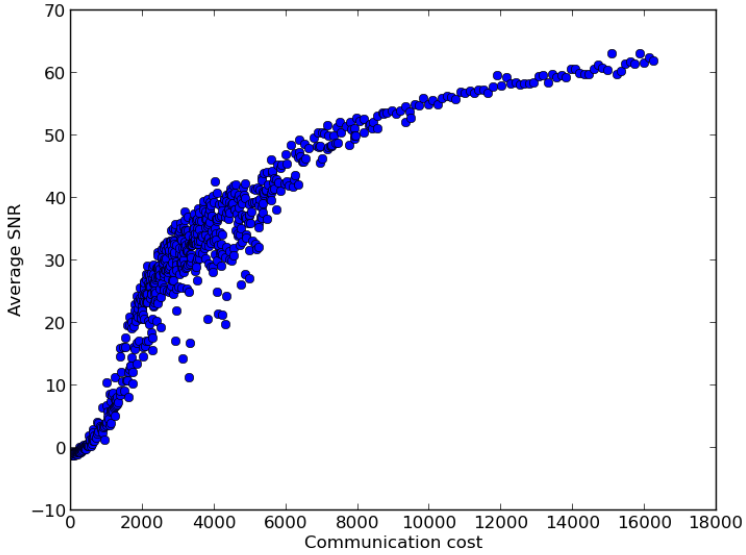


Figure A.12: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 12

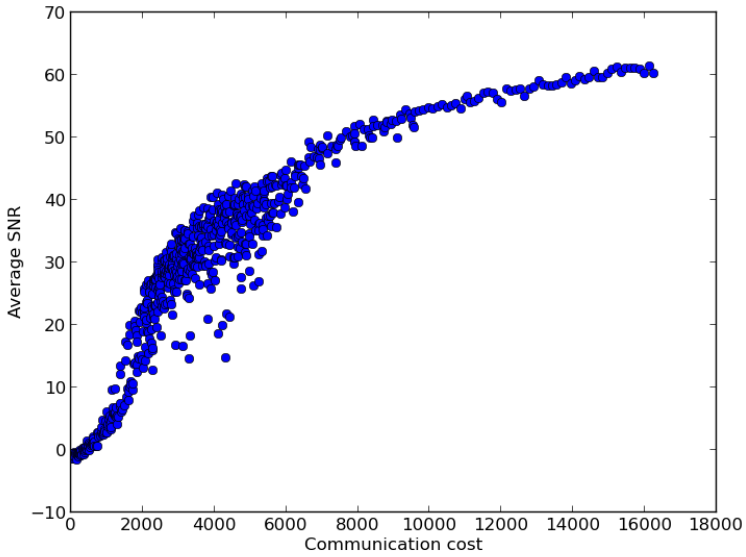


Figure A.13: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 13

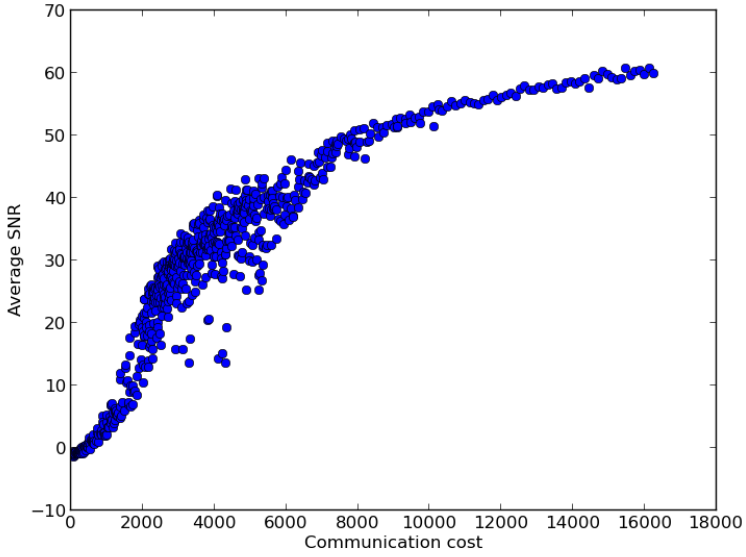


Figure A.14: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 14

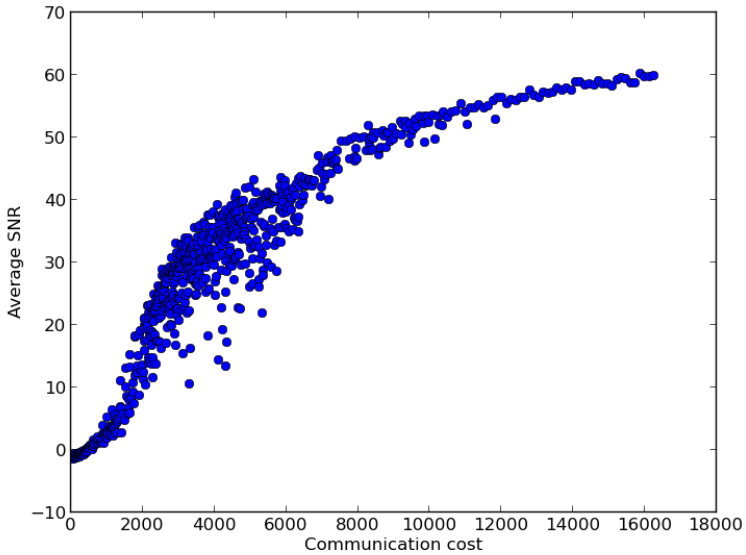


Figure A.15: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 15

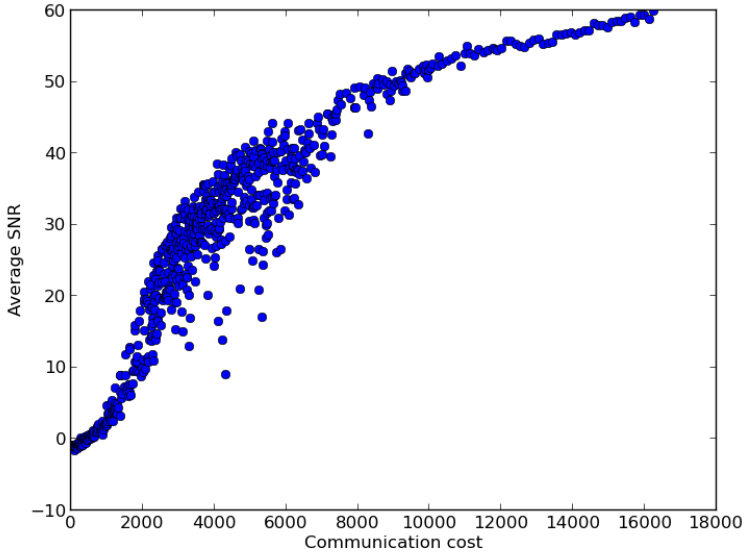


Figure A.16: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 16

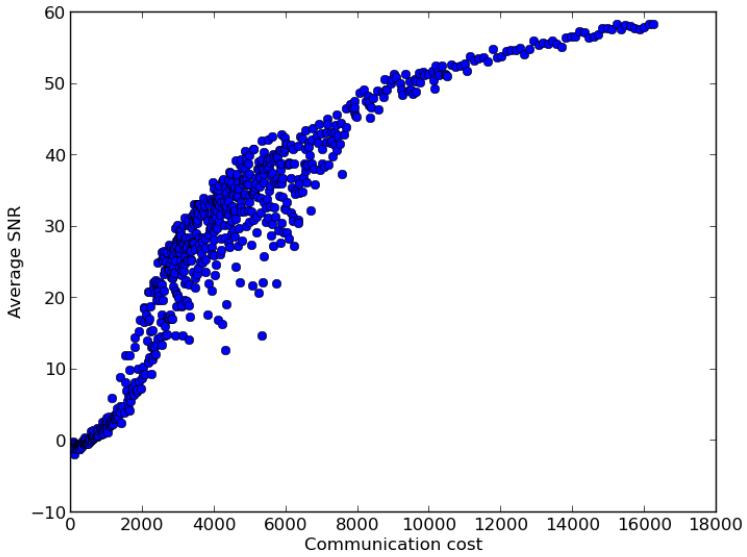


Figure A.17: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 17

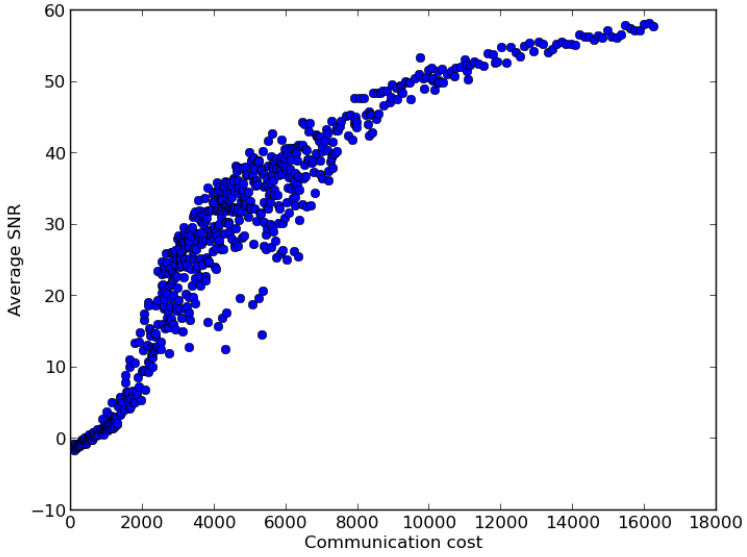


Figure A.18: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 18

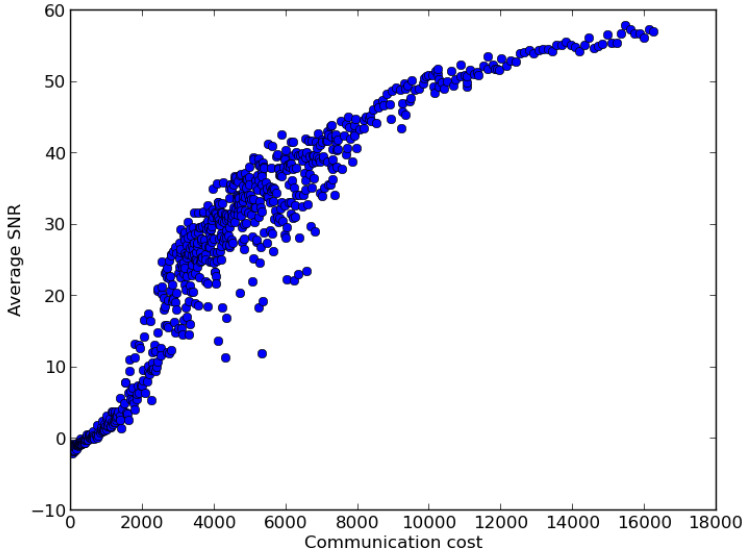


Figure A.19: Average signal accuracy vs. communication cost when sparsity parameter s is equal to 19

Bibliography

Sensorscope LUCE deployment at École Polytechnique Fédérale de Lausanne, 2008. URL `http://lcav.epfl.ch/op/edit/sensorscope-en`.

I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002. ISSN 1389-1286.

W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak. Joint source-channel communication for distributed estimation in sensor networks. *IEEE Trans. on Information Theory*, 53(10):3629–3653, 2007.

W. Bajwa et al. Compressive wireless sensing. In *Proc. of the 5th international conference on Information processing in sensor networks (IPSN)*, pages 134–142, 2006.

Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27 (3):265–274, 2009.

S Allen Broughton and Kurt M Bryan. *Discrete Fourier analysis and wavelets: applications to signal and image processing*. John Wiley & Sons, 2011.

C. Caione, D. Brunelli, and L. Benini. Compressive sensing optimization over zigbee networks. In *Proc. of the International Symposium on Industrial Embedded Systems (SIES)*, pages 36–44, 2010.

E. J. Candes and Y. Plan. A probabilistic and ripless theory of compressed sensing. *IEEE Trans. on Information Theory*, 57(11):7235–7254, 2011. ISSN 0018-9448.

E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. on Information Theory*, 52(12):5406 –5425, 2006.

E. J. Candes et al. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. on Information Theory*, 52(2):489 – 509, 2006a.

Emanuel J. Candes. Compressive sampling. In *Proc. of the International Congress of Mathematicians*, volume 3, pages 1433–1452, 2006.

Emmanuel Candes and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969, 2007.

Emmanuel J. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589 – 592, 2008. ISSN 1631-073X.

Emmanuel J. Candes, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006b.

K. Chintalapudi, T. Fu, Jeongyeup Paek, N. Kothari, S. Rangwala, J. Caffrey, R. Govindan, E. Johnson, and S. Masri. Monitoring civil structures with a wireless sensor network. *IEEE Internet Computing*, 10(2):26–34, 2006.

R. Coifman, F. Geshwind, and Y. Meyer. Noiselets. *Applied and Computational Harmonic Analysis*, 10(1):27 – 44, 2001. ISSN 1063-5203.

Joachim Dahl and Lieven Vandenberghe. CVXOPT: A python package for convex optimization, http://www.abel.ee.ucla.edu/cvxopt, 2006.

D. L. Donoho, M. Elad, and V.N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. on Information Theory*, 52(1):6 – 18, 2006.

David Donoho and Jared Tanner. Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4273–4293, 2009.

D.L. Donoho. Compressed sensing. *Information Theory, IEEE Trans. on*, 52(4):1289 –1306, 2006. ISSN 0018-9448.

M. F. Duarte, M. B. Wakin, D. Baron, and R.G. Baraniuk. Universal distributed sensing via random projections. In *Proc. of the 5th international conference on Information processing in sensor networks (IPSN)*, pages 177–185, 2006.

M.F. Duarte et al. Distributed compressed sensing of jointly sparse signals. In *Conference Record of the 39th Asilomar Conference on Signals, Systems and Computers*, pages 1537 – 1541, 2005.

T. ElBatt. On the trade-offs of cooperative data compression in wireless sensor networks with spatial correlations. *IEEE Tran. on Wireless Communications*, 8(5):2546–2557, 2009. ISSN 1536-1276.

Jeremy Elson and Deborah Estrin. Sensor networks: A bridge to the physical world. In C. S. Raghavendra, Krishna M. Sivalingam, and Taieb Znati, editors, *Wireless Sensor Networks*, pages 3–20. Springer US, 2004. ISBN 978-1-4020-7884-2.

Antonio Franchi, Giuseppe Oriolo, and Paolo Stegagno. Mutual localization in a multi-robot system with anonymous relative position measures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009*, pages 3974–3980, 2009.

M. Gastpar, P. L. Dragotti, and M. Vetterli. The distributed karhuenen-loève transform. In *IEEE Transactions on Information Theory*, volume 52, pages 5177–5196, 2006.

Amara Graps. An introduction to wavelets. *IEEE Compututer Science Engineering.*, 2:50–61, June 1995.

J. Haupt and R. Nowak. Signal reconstruction from noisy random projections. *IEEE Transactions on Information Theory*, 52(9):4036–4048, 2006. ISSN 0018-9448.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, http://www.scipy.org, 2001–.

J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for smart dust. In *Mobicom'99*, MobiCom '99, pages 271–278. ACM, 1999. ISBN 1-58113-142-9.

S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior point method for large-scale l1-regularized least squares. *Selected topics in Signal Processing, IEEE Journal*, 1(4):606–617, 2007.

C. Luo, F. Wu, J. Sun, and C. Chen. Efficient measurement generation and pervasive sparsity for compressive data gathering. *IEEE Trans. on Wireless Communications*, (99):1–11, 2010.

Chong Luo et al. Compressive data gathering for large-scale wireless sensor networks. In *Proc. of the 15th annual international conference on Mobile computing and networking (Mobicom)*, pages 145–156, 2009.

S. Mahfoudh and P. Minet. Survey of energy efficient strategies in wireless ad hoc and sensor networks. In *Proc. of the Seventh International Conference on Networking*, pages 1–7, 2008.

M. Mahmudimanesh, A. Khelil, and N. Suri. Balanced spatio-temporal compressive sensing for multi-hop wireless sensor networks. In *IEEE MASS'12*, pages 389–397, 2012.

Antonio Ortega Marco F. Duarte, Godwin Shen and Richard G. Baraniuk. Signal compression in wireless sensor networks. *Philosophical Trans. of the Royal Society*, 370(1958):118–135, 2012.

Harald Niederreiter. Recent trends in random number and random vector generation. *Annals of Operations Research*, 31(1):323–345, 1991. ISSN 0254-5330. doi: 10.1007/BF02204856. URL `http://dx.doi.org/10.1007/BF02204856`.

Olfati-Saber et al. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2006. ISSN 0018-9219.

P. M. Parekar and S. S. Thakare. Lossless data compression algorithm–a review. *International Journal of Computer Science & Information Technologies*, 5(1), 2014.

J. Polastre et al. Telos: enabling ultra-low power wireless research. In *ACM IPSN'05*, pages 364–369, 2005.

M. Rabbat, J. Haupt, A. Singh, and R. Nowak. Decentralized compression and predistribution via randomized gossiping. In *The 5th Intl. Conf. on Information Processing in Sensor Networks, IPSN 2006.*, pages 51–59, 2006.

Paolo Santi. Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194, June 2005. ISSN 0360-0300.

G. Shen and A. Ortega. Transform-based distributed data gathering. *IEEE Trans. Signal Processing*, 58(7):3802–3815, 2010.

D. Slepian and J.K. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. Inform. Theory*, 19(4):471–480, 1973.

K. Sohraby, D. Minoli, and T. Znati. *Wireless Sensor Networks, Technology, Protocols and Applications*. Wiley-Interscience, 2007. ISBN 9780471743002.

Tossaporn Srisooksai, Kamol Keamarungsi, Poonlap Lamsrichan, and Kiyomichi Araki. Practical data compression in wireless sensor networks: A survey. *Journal of Network and Computer Applications*, 35(1):37 – 59, 2012. ISSN 1084-8045.

J. A. Stankovic, T.F. Abdelzaher, Chenyang Lu, Lui Sha, and J.C. Hou. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7):1002–1022, 2003.

F. Stann and J. Heidemann. RMST: reliable data transport in sensor networks. In *Proc. of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 102 – 112, 2003.

Hüseyin Özgür Tan and Ibrahim Körpeoğlu. Power efficient data gathering and aggregation in wireless sensor networks. *SIGMOD Rec.*, 32(4):66–71, December 2003. ISSN 0163-5808. doi: 10.1145/959060.959072. URL `http://doi.acm.org/10.1145/959060.959072`.

J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. on Information Theory*, 53 (12):4655 –4666, 2007.

R. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer-Verlag, 2001.

Roberto Verdone, Davide Dardari, Gianluca Mazzini, and Andrea Conti. *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*. Academic Press, 2008. ISBN 0123725399, 9780123725394.

Mehmet C. Vuran, A. B. Akan, and Ian F. Akyildiz. Spatio-temporal correlation: Theory and applications for wireless sensor networks. *Computer Networks*, 45(3):245 – 259, 2004. ISSN 1389-1286.

Michael B Wakin, Marco F Duarte, Shriram Sarvotham, Dror Baron, and Richard G Baraniuk. Recovery of jointly sparse signals from few random projections. In *NIPS*, 2005.

S. Winter, H. Sawada, and S. Makino. On real and complex valued l1-norm minimization for overcomplete blind source separation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 86 – 89, 2005.

A. Wyner and J. Ziv. The rate-distortion function for source coding with side information at the decoder. *IEEE Trans. Inform. Theory*, 22(1):1–10, 1976.

Han Lun Yap et al. The restricted isometry property for block diagonal matrices. In *Proc. of the 45th Annual Conference on Information Sciences and Systems (CISS)*, pages 1 –6, 2011.