Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Steve Saed

Entitled Average Consensus in Wireless Sensor Networks with Probabilistic Network Links

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Lingxi Li
Chair

Dongsoo S. Kim

Brian King

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Lingxi Li

Dongsoo S. Kim

Approved by: Brian King 07/07/2010
Head of the Graduate Program Date

Graduate School Form 20
(Revised 1/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:
Average Consensus in Wireless Sensor Networks with Probabilistic Network Links

For the degree of Master of Science in Electrical and Computer Engineering

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Teaching, Research, and Outreach Policy on Research Misconduct (VIII.3.1)*, October 1, 2008.*

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Steve Saed
Printed Name and Signature of Candidate

07/07/2010
Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/viii_3_1.html

AVERAGE CONSENSUS IN WIRELESS SENSOR NETWORKS WITH PROBABILISTIC NETWORK LINKS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Steve Saed

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2010

Purdue University

Indianapolis, Indiana

ACKNOWLEDGMENTS

I would like to acknowledge my thesis advisors, Dr. Dongsoo S. Kim and Dr. Lingxi Li, who generously shared with me their time, experience and knowledge, for their guidance, supervision and support throughout my career as a master's student.

Also, I would like to extend my special thanks to Dr. Brian King who served as a member of my advisory committee and who also assisted me in my research.

Finally, I would like to thank my family for their love and support throughout my career as a university student.

TABLE OF CONTENTS

Page

LIST OF FIGURES

ABBREVIATIONS

WSN	Wireless Sensor Network
MANET	Mobile Ad Hoc Network
ADC	Analog to Digital Converter
RF	Radio Frequency
ITS	Intelligent Transport System
AF	Approximated Fading

ABSTRACT

Saed, Steve. M.S.E.C.E., Purdue University, August 2010. Average Consensus in Wireless Sensor Networks with Probabilistic Network Links. Major Professors: Dongsoo S. Kim and Lingxi Li.

This study proposes and evaluates an average consensus scheme for wireless sensor networks. For this purpose, two communication error models, the fading signal error model and approximated fading signal error model, are introduced and incorporated into the proposed decentralized average consensus scheme. Also, a mathematical analysis is introduced to derive the approximated fading signal model from the fading signal model. Finally, different simulation scenarios are introduced and their results analyzed to evaluate the performance of the proposed scheme and its effectiveness in meeting the needs of wireless sensor networks.

1. INTRODUCTION

In recent years, major advances in wireless communication and digital electronics led to the development of tiny sensors that can broadcast sensed data over short communication ranges. A wireless sensor network (WSN) is composed of a number of sensor nodes that have sensing, processing and communication components that enable all the network nodes to collaborate in order to achieve a particular task [1]. Each node in a WSN is designed to have limited computational and power capacities and use basic broadcast communication protocols in order to exchange information with its neighbors [2]. Moreover, sensor nodes are designed to be densely and randomly deployed very close to the phenomenon that the WSN is expected to monitor. This allows WSNs to be used in a wide range of applications where the exact position of each sensor cannot be pre-determined and the nature of the topology makes more sensors prone to failure such as the ones used in military applications [3].

Wireless sensor networks gained the attention of many research communities due to the wide range of applications where they can be employed. Military applications of WSNs include battlefield surveillance where WSNs deployed in enemy territory can track and report the movement of enemy troops. Another important military application is detecting chemical and biological attacks by monitoring the atmosphere of the battlefield [1]. Furthermore, WSNs can be used in many different environmental applications that intend to track the movement of certain species in remote ecosystems, detect forest fire, monitor the pollution at the bottom of a certain river, [4], [5], [6], [7]. Furthermore, WSNs can be used in health-care, home-applications [8], [9] [10], etc.

Several applications of WSNs can be incorporated in transportation systems. First, WSNs can be integrated into vehicular ad hoc networks (VANETs), which are a type of ad hoc wireless mobile networks used to exchange information among

vehicles on the road [11]. Sensors in each vehicle are used to collect information about road congestion, speed of the different vehicles on the road, etc, which is then transmitted from one vehicle to another. This information can be used to alert the drivers to potential hazards which would help improve road safety [12]. Also, such information can be used to reduce traffic congestions by giving drivers information about congestions ahead of them so that they find a less congested route to their destination [13]. Second, WSNs can be used in intelligent transportation systems (ITS), which are composed of a bundle of sensing, communication and decision-making components added to vehicles that are designed to help the driver avoid any potential hazards. For instance, an ITS mounted in a certain vehicle can monitor the distance from other vehicles on the road and automatically control vehicle-to-vehicle distance when necessary [14].

Many problems often arise in WSNs due to their network architecture, environments where they are deployed or the computational and power limitations of their nodes. The network nodes are often exposed to harsh conditions, like the ones experienced in a battlefield, or may run out of power after a short period of time, which means that the network is going to be losing nodes over time. Also, the modest computational power of the nodes will impose certain constraints on the algorithms that are going to be implemented in the network in terms of computational requirements, memory storage capabilities and computational efficiency. Consequently, the algorithms implemented in the network need to work in a decentralized manner in order to minimize the effect of losing nodes on the network functionality. Other problems experienced in WSNs are similar to those found in all ad hoc mobile wireless networks like communication errors, hidden terminal problem, exposed terminal problem, etc.

In many applications where WSNs are used, finding the average of a certain measurement among all the nodes in the network is very common [15], [16], [17], [18], [19], [20], [21]. Such a problem is often described as a consensus problem, because it involves reaching an agreement regarding a certain quantity of interest that depends on the state of all agents [22]. This is essentially useful in applications where the

network needs to make decisions regularly about a certain measurement it takes. For example, a WSN can be deployed to detect the concentration of a certain chemical in the atmosphere. Reaching an average consensus among the network nodes on all the initial measurements helps spread a reasonable estimation of the value measured to all parts of the network which may have not made equally robust measurements due to hardware faults, location of the different nodes in the topology where the network is deployed, etc.

This study introduces a new average consensus scheme that takes into consideration the constraints of WSNs in terms of communication errors. The consensus algorithm and the weighting scheme used in this study were adopted from previous studies on this topic [23] because of their effectiveness in achieving average consensus in a decentralized manner, thus overcoming some of the constraints found in WSNs as mentioned earlier. Several simulations were carried out to illustrate the proposed consensus schemes.

Furthermore, two wireless communication error models are introduced. The first one is the fading signal error model and the second is the approximated fading signal error model, where the latter is derived from the former. Also, mathematical analysis in introduced to support this derivation and its significance is discussed in terms of different factors relevant to the problem introduced.

The outline of this study is as follows: Chapter 2 summarizes some of the related studies and sheds light on their contributions and shortcomings. Some graph theory concepts are introduced in chapter 3. Chapter 4 describes the problem definition and the proposed average consensus scheme. Simulations carried out to evaluate the performance of the scheme are described in Chapter 5 and the simulation results are analyzed in Chapter 6. Finally, Chapter 7 concludes this study.

2. GRAPH THEORY OVERVIEW

A graph $G(V, E(t))$ is defined as a 2-tuple consisting of a vertex set V and an edge set $E(t)$ [24]. In this study, the vertices in G represent the sensors of the WSN . The presence of an edge between any two given vertices i and j in G indicates the presence of a communication link between the sensors represented by the respective nodes.

The edge set $E(t)$ is considered to be dynamic as links can be formed or broken between nodes over time.

The graph G is assumed to be directed as full duplex communication between any two given nodes in a WSN is not very likely due to channel noise and communication failures.

The graph $G(V, E(t))$ has a corresponding $N \times N$ adjacency matrix defined as [24]:

$$A(i,j) = \begin{cases} 1 & \text{if}(i,j) \in E(t), \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

The neighborhood, $\mathcal{N}_i$, of a given vertex i is defined as the set of vertices that are within communication range of vertex i such that communication links are formed between i and each vertex $j \in \mathcal{N}_i$.

The degree of a node d_i is the number of edges that exist between vertex i and its neighbors, $\mid \mathcal{N}_i \mid$ [24].

The degree matrix of a graph $G(V, E(t))$ is an $N \times N$ matrix defined as [24]:

$$\boldsymbol{D} = \text{diag}(d_1, ..., d_N). \tag{2.2}$$

The Laplacian $\boldsymbol{L}$ of a matrix is defined as [24]:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{2.3}$$

Alternatively, it can be defined as follows:

$$L(i,j) = \begin{cases} \deg(V_i) & \text{if } \ i = j, \\ -1 & \text{if}(i,j) \in E(t), \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

The Laplacian is a positive semi-definite symmetric matrix and hence its eigen-values are all non-negative:

$$0 = \lambda_1(\boldsymbol{L}) < \lambda_2(\boldsymbol{L}) \leq ... \leq \lambda_N(\boldsymbol{L}) \tag{2.5}$$

The number of connected components in $G(V, E(t))$ is equal to the multiplicity of the zero eigen-values of $\boldsymbol{L}$ [25]. Also, The *algebraic connectivity* of a graph is defined as the second smallest eigen-value of the $\boldsymbol{L}$ matrix, $\lambda_2(\boldsymbol{L})$. The *spectral radius* of $\boldsymbol{W}$, the weight matrix of the graph, is defined as absolute value of the second largest eigen-value, of the matrix. The value of $\lambda_2(\boldsymbol{L})$ is greater than zero when the corresponding graph is connected.

A graph is considered to be strongly connected if there exists a simple path between any two given vertices in the graph. The graph $G(V, E(t))$ is considered to be strongly connected if rank($\boldsymbol{L}$) = N-1 [17].

The edge set *E(t)* of the graph $G(V, E(t))$ can be weighted, which means that there is a certain weight value associated with every edge. A directed graph would be considered balanced if and only if the total weight entering any given vertex $i \in V$ is equal to the total weight leaving this vertex. An undirected graph is balanced by default.

3. RELATED WORK

3.1 Wireless Sensor Networks

As mentioned before, a wireless sensor network (WSN) is a network composed of a number of sensor nodes that have sensing, processing and communication components that enable all the network nodes to collaborate in order to achieve a particular task [1].

One of the most important features of WSNs is that the sensor nodes need not to be placed in any specific position in the topology in order to achieve the desired task. The nodes are randomly deployed all over the topology where the algorithms and protocols running in the network are expected to run without having any specific knowledge about the topology. This has secured a wide range of applications for WSNs [1].

Wireless Sensor Networks are a subset of Ad Hoc Wireless Mobile Networks (MANETs). This is because the algorithms and protocols of WSNs are expected to run without having any network infra-structure such as cellular towers. However, there exists a number of differences between WSNs and MANETs summarized as follows [2]:

- The number of nodes in a WSN is usually much larger than the number of nodes in MANETs. This is because nodes in a WSN are expected to monitor a certain phenomenon from many different angles and directions.
- Consequently, nodes in WSNs are more densely deployed than nodes in MANETs.
- Nodes in WSNs usually have more limited computational and power capacity. This is because a WSN is expected to have a large number of nodes. Thus,

reducing the quality of the components of each node will reduce its price and help control the overall cost of the WSN.

- Nodes in WSNs are expected to operate in rough environments where the nodes could get damaged or even stop functioning over time. This is not always the case in MANETs.

- As a result, nodes in WSNs are more likely to fail during the operation of the network, and hence the topology of WSNs is expected to change over time.

Any given sensor node is expected to at least have the following four components [1]:

- *Sensing unit*: this is composed of a sensor and an analog to digital converter (ADC). The sensor measures a certain metric of interest in the phenomenon that the WSN is supposed to monitor as an analog signal. Then, the ADC will convert this to a digital signal for the node's processor to work with.

- *Processor*: this is the unit that runs the WSN protocols and algorithms on the data that it receives from the sensing unit to achieve the task that the WSN is assigned. The processor only has access to a small memory space.

- *Transceiver*: this connects the node to the rest of the WSN. It is assigned the task of broadcasting the processed data and receiving other nodes' data.

- *Power unit*: this usually includes a power source such as a battery and a power management unit that attempts to reduce the consumption of power when the node is idle. Some nodes may contain a power generator that may run on solar power or some other means to recharge the node's power source.

In addition to the main components listed above, the node may also contain a "location finding system", which is assigned the task of figuring out some location information for the node. This is mostly common in mobile WSNs as the node may

need to know the distance it moved or the direction it moved in. Many other metrics could be relevant depending on the application that the WSN is assigned to.

Given the aforementioned hardware features of WSNs, there is a number of design constraints that a typical WSN is expected to consider [26]:

- *Low power consumption*: this is one of the most important constraints that the hardware design of the WSN nodes must meet. Thus, the WSN should be designed in such a way that minimizes data communication and processing. Furthermore, the hardware components should be chosen to have the minimum number of features required for the operation of the WSN. The presence of extra unneeded components may lead to power drain.

- *Environment-adaptiveness*: the environments where WSNs deployed are usually rough. Thus, the node must be designed in such a way that allows it to be more resilient to collisions with hard surfaces, water, sand, etc, depending on the nature of the topology where the network is deployed.

- *Autonomy*: the WSN node should not require any network infra-structure or fusion node for proper operation.

- *Low production cost*: WSNs are expected to be composed of a large number of nodes. Certain applications may require several thousands of them. As a result, each individual node needs to be produced at a low cost, which means that the designers might need to make certain trade-offs with the quality and resilience features mentioned previously.

A study [27] suggested that the cost of the individual node needs to be less than one US dollar for the cost of the WSN to be justifiable. This is very hard to achieve given the different components that need to be present in a WSN node:

- The power unit of the WSN node needs to last for a significant period of time. Furthermore, it needs to be small enough to fit the size of the WSN node. In

some applications, the power unit may be equipped with a device to collect solar energy to recharge the power unit [28].

- The transceiver unit of the WSN node needs to operate radio frequency (RF) communication as the size of the data communicated in a WSN is relatively small. RF communication requires modulation, demodulation, multiplexing, etc, which all requires expensive circuitry.

- Many application require the node to have a certain location-finding device such as a GPS. Equipping each node in the network with a GPS will definitely make the cost per node to be higher than one US dollar. It was suggested that each group of nodes share one GPS device mounted on one of them to reduce the cost [29].

Another challenge that often arises in WSNs is topology management. This is because the nodes in a WSN are prone to failure, power depletion, etc, which makes the topology of the network dynamic. Moreover, nodes are often deployed in very large numbers within a relatively small topology, which can make certain nodes isolated from the rest of the network because of lack of opportunity to receive or broadcast data to the rest of the network due to high network activity [1].

There are certain phases during the lifetime of the WSN where the topology needs to be carefully managed, as otherwise the WSN may fail at carrying out the task it was assigned [1]. The first phase occurs before the deployment of the network. Usually, the nodes are randomly scattered all over the intended topology of the WSN. While nodes have no predefined location to be placed in, scattering the nodes outside the topology, far away from the phenomenon that the WSN is supposed to monitor, will reduce the efficiency of the network. The second phase occurs right after the deployment of the network as this is the time when faulty nodes and other problems will appear. The WSN should be designed to recover from the impact of such problems. The third phase occurs after a significant period of time from the deployment of the network when the WSN would need to be re-organized in order to make up for the effect

of losing nodes because of power depletion, jamming, isolation, etc. The WSN is expected to have recovery mechanisms built-in to account for all three phases.

The management of power consumption is one of the greatest difficulties associated with designing algorithms and protocols for WSNs [1]. A typical node in a WSN has very limited power resources. If the WSN is used in a military application for example, like remote monitoring of enemy troops' movement, it would be impossible to recharge the power of the nodes. Thus, nodes are prone to power depletion over time. Moreover, nodes in multi-hop WSNs act not only as transmitters and receivers of data, but also as data routers. Many network algorithms count on the nodes ability to move data from one node to another through a series of other nodes because of the absence of network infra-structure that usually plays this role. Hence, nodes running out of power will have a double impact as their monitoring and processing power is lost as well as their routing of data to different parts of the WSN.

Consequently, algorithms and protocols that run on the WSN need to be power-aware [1]. The design of such algorithms need to give power consumption top priority, where trade-offs with other factors such as packet delay and throughput will often be made. For instance, common ad hoc network routing algorithms will choose the path that minimizes packet delay, the time between the sender transmitted the packet and its reception at its destination. Also, such algorithms may pick the route where most throughput will most likely be achieved, that is where the largest amount of data will pass through in a given period of time. In a WSN, the route chosen will also need to consider the power levels in each node along the route chosen in such a manner that it does not deplete power in any node; otherwise overloaded nodes will run out of power and the topology of the network will deteriorate affecting the performance of the WSN.

In a WSN, each node carries sensing, processing and communication tasks. Thus, power consumption can be examined separately for each task [1]:

- *Sensing*: the power-consumption of the sensing task is application and environment dependent. In certain environments, there exists a high noise level that

makes it hard to make accurate measurements. Thus, the sensing unit of the node will need to employ different filtering mechanisms in order to improve the accuracy of the measurements it makes. Such mechanism could be as simple as averaging out several measurements, but others could use advanced power-thirsty algorithms. Furthermore, certain applications where delicate changes in the environment need to be detected, the sensing unit will need to have more advanced capabilities that are most likely going to consume significantly more power than applications where the node does not have to be extremely accurate.

- *Processing*: this is lowest consumer of power in a WSN node. Thus, it would make the whole WSN more efficient in terms of power consumption to have WSN algorithms designed in such a way that allows the data sensed to be processed in the same node. As it will be discussed later, communication consumes significantly more power than processing. As a matter of fact, most processing in a WSN network is most likely going to be arithmetic operations on scalars, which does not need that much computational power and hence does not consume much power. Moreover, the limited cost and size of the node imposes CMOS (complementary metal oxide semiconductor) micro-processor technology on the design of WSN nodes. The transistors of this technology consume more power when it is switched on than those made using different technologies. Thus, lowering the supply voltage to the circuitry of the processor will help reduce the inefficiency introduced by the CMOS technology. In addition, certain applications may require additional circuitry such as a decoder or an encoder which may increase the power consumption of the processing circuitry.

- *Communication*: This consists of data transmission and reception. Most of the power in a WSN node is expected to be spent by the communication unit. Communication over long ranges is more power consuming than short range communication. Thus, the nodes of WSNs have short communication ranges in order to save power. Moreover, the start-up of the communication circuity

is a major power consumer due to the design of the circuitry associated with the transceiver. Since WSNs don't usually communicate a large volume of data packets, the start-up power consumption is dominant in the communication unit. Thus, the start-up circuity is never turned off as this will become a major power inefficiency. Some studies such as [30] proposed a low power transceiver architecture.

As a result of the different constraints that often arise in WSN, the protocol stack of the WSN is designed in such a way that minimizes the consumption of power, builds the necessary infrastructure for communication with other nodes via the transceiver unit of the WSN node and allows different kinds of algorithms and protocols run in the WSN. The protocol stack usually consists of the following layers [1]:

- *Application layer*: this layer is application dependent. It is sometimes useful to include certain application level protocols that can, for instance, help manage the data flow in the network. However, this layer seems to be one of the least explored fields in WSN-related topics.

- *Transport layer*: this layer is important when the WSN needs to connect to other networks such as the internet for example.

- *Network layer*: this layer contains protocols that are mostly associated with data-routing. Certain aspects of the power consumption issue can be dealt with in this layer by introducing routing algorithms that take power consumption in each node into consideration when making decisions in multi-hop communication.

- *Data link layer*: data frame detection, error control and data stream management are among the tasks that are carried out in this layer.

- *Physical layer*: this layer is usually assigned different tasks among which are data encryption, frequency generation, modulation and signal detection.

3.2 Applications of Wireless Sensor Networks

There exists many different types of sensors that can be used in WSNs in order to monitor a variety of phenomena in different environments for a wide range of applications. The following is a partial list that includes a number of scalar quantities that can be sensed by a WSN for different applications [31]:

- *Temperature*: one possible application is monitoring forest-fire.
- *Pressure*:one possible application is monitoring movement of enemy troops and military vehicles in the battlefield.
- *Vehicular speed*:one possible application is Vehicular Ad Hoc Networks (VANETs) where sensors mounted on vehicles are try to estimate the speed of neighboring vehicles.
- *Noise level*:one possible application is monitoring traffic movement.
- *Lightning conditions*:one possible application is monitoring hazardous weather conditions in remote areas.
- *Soil makeup*:one possible application is monitoring pollution in the environment.

As a result, many different applications of WSNs were devised. One of the most important fields that took advantage of the many features that WSNs provide is Intelligent Transport Systems (ITS) [32–35].

Intelligent transport systems are systems concerned with the management of traffic. They are designed to deal to problems such as traffic congestion in urban areas using a variety of equipment that monitor the traffic then use the data collected in order to prevent traffic jams [35].

Traditional ITSs used very expensive, power-thirsty equipment such as ultrasonic sensors, video-cameras, etc to monitor traffic. The hardware used in ITSs is usually connected using cables and requires all the data accumulated to be sent to a fusion

center or a central computer to be processed. This limits the installation of ITSs to areas where it is possible to build the necessary cable lines and where communication with an external network such as the internet exists. The equipment will need to be accessed regularly in order to charge their power sources [36].

Integrating WSNs with low-power nodes into ITSs will boost the ability of the different types of equipment used in ITSs to collaborate and make decisions in a decentralized manner. Furthermore, it reduces deployment and maintenance costs and helps remove network infra-structure, which WSNs can work in its absence, from the road. This will enhance the ability of of ITSs to help drivers reduce travel time by reducing traffic congestions [37].

The sensors used in ITSs usually fall under two different categories [38]:

- *Intrusive sensors* are sensors installed on the road's surface, inside openings in the road or underneath the road's surface. They are considered to be highly accurate as their technology is deemed to be mature and their deployment and operation is widely understood. On the other hand, intrusive sensor tend to have a number of disadvantages such as high cost of installation, deployment and maintenance as the traffic need to be stopped in order to carry out such activities. Performing maintenance on the roads where such sensors are installed means that the sensors will have to be removed and re-installed. Consequently, the sensors lifetime is correlated with the condition of the road itself as poor roads will shorten their lifetime and road maintenance could possibly damage them or hinder their operation.

- *Non-intrusive sensors* are sensors mounted above ground level against the flow of traffic direction in order to monitor vehicles moving in different lanes. Such sensors include video cameras, passive acoustic array, Doppler or infrared sensors. Non-intrusive sensors count the number of vehicles, their speed and other information that may be useful in traffic management. The advantage of using non-intrusive sensors is that they don't require much effort to install. They can

be mounted on the side of the road without any problems. On the other hand, they usually require a lot of power to operate, perform poorly in bad weather conditions such as fog or snow, and still require a fusion center to process the data they collect in order to make decision on traffic management.

Neither the intrusive nor the non-intrusive sensors can be used in WSNs as they are power-thirsty and lack the ability to process and communicate the data they sense. As a result, more advanced, smaller and low power sensors were developed [35] that fit the WSN better. Introducing RF communication, smaller sensor-size, network algorithms and power efficiency will allow the sensors of the WSN to be installed like the intrusive sensors, in the road or underneath it. The communication and processing abilities added to the WSN protocols and algorithms will allow the network to make decisions without the need to send the sensed information to a fusion center. Furthermore, the small size will make installing the sensors quicker and easier as there is no need to dig into the road in order to install the sensors. Also, the enhanced power efficiency means that the batteries will not need to be changed very often. A study [35] predicts that the sensors' battery might last up to 10 years. Moreover, using the smaller in-pavement sensors will help collect very accurate information about the vehicle, which is the main advantage of intrusive sensors.

In order to integrate WSNs into ITSs properly, the network algorithms designed for such applications need to take into consideration the following constraints [35]:

- In case some nodes of the WSN run out of power or get damaged, the network algorithms designed need to be able to continue their operation without much affecting the performance of the application. Hence, the algorithms need to work in a decentralized manner, where each node operates on its own without any consideration for other nodes in the network or any topology changes.

- Several network algorithms need to be able to run concurrently. Hence, several communication protocols will need to be developed in order to distinguish different types of data and the purpose it was communicated for. This can be

implemented as headers attached to data packets that contain some identification information.

- The network algorithms designed for such a system need to be power aware in order to maximize the lifetime of the system. Thus, the nodes need to be able to switch to hibernation mode when no vehicles are spotted. It is important not to completely turn off the nodes as the circuitry that turns them on consumes a significant amount of power.

- Focussing on power means that the network algorithm designed is not going to perform optimally in terms of throughput and packet delay. Lowering the network throughput is also going to lower the use of the network communication channels. Furthermore, it is going to be assumed that the nodes are not going to be mobile, which will reduce the expected consumption of power per node [39, 40].

- Reducing data communication will also reduce the power consumption. One study [37] proposed using a data compression scheme that could minimize the size of the data communicated. This scheme is based on comparing the current data with historical data. If they are highly correlated, then a small portion of the fresh data is sent. However, using this scheme means that the processor of the WSN node needs to have some advanced processing and storage capabilities, which WSN nodes are not expected to have.

The integration of WSNs into ITSs has many applications that could make traffic management a much smoother task in a variety of topologies where traffic needs to be managed. One such topology is a parking lot [41]. Large malls, stadiums, airports, etc have a large number of parking spaces where many vehicles are trying to park. However, due to the compact design of those parking lots, many drivers find themselves driving in circles trying to find a parking spot. Thus, WSNs and ITSs can be used in parking lots in order to make it easier for drivers to find parking spots. This can be done by placing a sensor at one end of each parking spot in the lot. If the

spot is available, the sensor will broadcast some information about the availability of the spot. Then, a decentralized network algorithm will determine the neighborhoods of sensors that have the largest numbers of available spots and send this information to a traffic management center that will take care of directing the drivers to those neighborhoods. In order to reduce the power consumption of in the WSN, each node will broadcast its data at a rate of once every ten minutes. This rate can be improved as data collected from the parking lot, where the WSN is deployed, is saved. Then, historical data accumulated can be used to build trends that can be used to better manage the parking lot.

Another application where WSNs and ITS are useful is monitoring traffic on roads and highways for congestion, hazardous drivers and other threats that could make the road less safe [36]. As discussed before, systems that carry out such tasks so far simply consist of video-cameras and other surveillance equipment that are wired to traffic management centers. The data sent from the surveillance equipment is then analyzed by the staff working at the management center. This system is very expensive as the hardware it uses is expensive and its power consumption is very high. Furthermore, this system cannot make its own decisions as it needs the staff that operates it to make decisions on traffic management.

Deploying WSNs in roads with high traffic volume, such as intersections [42], will enable collecting accurate information about the vehicles on those roads. The embedded sensors in the road will collect information about the vehicles' count, speed, motion direction, etc and employ certain network algorithms that will detect for instance cars moving at very high speeds or opposite to the direction that the traffic is supposed to move in. This data is then routed to access points mounted on the side of the road. The access points will broadcast the data to all vehicles on the road that can receive the alerts generated by the WSN using special network hardware designed for the ITS. If the drivers are informed early enough about the potential danger on the road, they can react in such a manner that protects them and the other drivers on the road.

The same system can run a different algorithm which will serve a different purpose on the road [42]. Certain intersections become very busy during rush hour and hence many drivers find themselves waiting a long period of time trying to go through the intersection from one side. However, the other side of the intersection might not be as busy, but the traffic light gives the same time period for cars in each side of the intersection to go through. Sensors embedded in the road can count the number of cars in each side of the intersection, then a network algorithm can increase the green time at the side with most traffic in order to reduce the congestion. This application can be further enhanced by placing devices on the cars themselves than can send information to access points on the road side, such as vehicle speed, size, etc, which can be later used by the network algorithm to make the decisions that will make the road safer and less congested.

Using WSNs integrated in ITSs will definitely improve traffic management. However, it would be very expensive to install a WSN on every single road in an entire metropolitan area. Thus, some algorithms can be designed to take in the data collected by the different WSNs deployed in a certain metropolitan area for instance to estimate the traffic flow in the roads between the WSNs [37]. This will help reduce the number of WSNs deployed but still manage the traffic in a larger number of roads.

Furthermore, the accumulated data can be used to build trends of traffic flow in different roads in a metropolitan area [43]. If the timing of the traffic congestion can be predicted using historic data collected over time, then measures can be taken in order to solve the problem permanently by building new roads.

3.3 Communication Error Model - Fading of Wireless Signals

Modeling errors in wireless communication channels is a very challenging task as it needs to take into consideration several different factors that cause errors in data transmitted over a wireless channel. Some of those factors are related to the environment where the wireless signal is transmitted and others are related to the

objects that reflect or diffract the signal. Thus, it is usually hard to generalize any mathematical analysis that will take into consideration every single factor that induces error in wireless communication. Consequently, many of the models that were constructed to describe the behavior of errors in wireless communication are faulty due to simplification or underestimation of factors that cause errors [44].

The following is a list of some of the most common causes that introduce error into the wireless signal:

- *Attenuation*: decrease of signal power at the receiver's end reducing the signal to noise ratio.
- *Doppler Shift*: If the sender and receiver are mobile, the difference in their velocity will make the reception of the signal harder.
- *Multi-path Fading*: fluctuation in angle, phase and amplitude of the signal due to the presence of obstacles between the sender and the receiver or due to other sources of error induced by the environment through which the signal is propagated.

As a result, introducing any wireless communication error model into the development of any network algorithm or protocol, such as the average consensus algorithm, is very challenging to achieve as it will most likely be able to satisfy only a few of the factors that cause error. On the other hand, considering no error model at all means that the algorithm assumes the free space model, where nothing but free space exists between the sender and the receiver, which is unrealistic in wireless communications [45]. Introducing a realistic error model will help better design and evaluate any network algorithm or protocol.

At the physical layer of the network, it is common to measure communication errors in Bit Error Rate (BER). However, most network algorithm and protocols that operate at the application layer of the network are mostly interested in a measure at the data packet level as most such applications exchange message that consist of a

series of packets, losing one of them will corrupt the whole message. Thus, in order to formulate an end-to-end measure of the performance of a given network protocol, it is important to conduct the error analysis at the packet level [44].

Furthermore, discrete time models such as the Markov Chain can be used to model the behavior of an entire communication link [46]. This is because errors usually occur in bursts which can be modeled by one state in a Markov chain and no errors can be modeled by another state. The communication link will move from one state to another depending on the error intensity.

One of the most important factors that lead to errors in wireless communication is fading. As mentioned before, it is caused by obstacles that stand in the way between the sender and the receiver causing the deflection of the wireless signal. This causes fluctuations in the signal's phase and angle.

The three main causes of error in radio propagation and hence wireless communication are [47, 48]:

- *Reflection* occurs when a signal falls on a surface and then gets reflected to a different direction where the angle of reflection is unpredictable. Furthermore, it is unpredictable if reflection will construct or destruct the signal.

- *Scattering* occurs when a radio signal hits an object that causes the signal to be dispersed in many different directions.

- *Diffraction* occurs when a signal falls on an object larger than the wave-length of the signal causing it to break up into a number of signals smaller in wave-length. While this introduces error into the signal, it can help receive the signal in areas with very large obstacles such as sky-scrappers.

The impact of fading increases on the strength of wireless signals at the receiver's end with respect to the propagation time and the distance between the sender and the receiver. As a result, many mathematical models were devised to describe fading in different scenarios [44]:

- One scenario is when the receiver node gets many scattered and reflected signals such that the overlapping signals cancel each other. It is proposed to use the Rayleigh distribution where the received power would be modeled as a random variable dependent on the distance between the sender and receiver.

- If the previous scenario contains some strong signal that stands out among all other signals, the Rice distribution may be used.

- If the scattering and reflection mentioned in the first scenario is severe, the Nakagami-m distribution may be used, as it can be used to model many different in-door and out-door conditions for both stationary and mobile nodes.

- If fading is accompanied by shadowing, the log-normal distribution may be used.

In addition to probability distribution functions, discrete-time modeling methods such as the Markov Chain can be used to describe the behavior of wireless communication channels. This model can be used to define two states of transmission, successful and unsuccessful. The transition probabilities associated with this model can be used to predict when the transmission is going to fail for each communication channel [49]. Other studies such as [50, 51] have extended this model to include certain forms of noise and predict channel quality.

Furthermore, the Markov model is very adapted to describing the behavior of communication channel for networks composed of mobile nodes [52]. Some of the most common metrics used in such cases are the link expiration time and the link connectivity [53, 54].

The link expiration time is defined as the period of time during which the connection between the sender and the receiver nodes is active. This is usually predicted in rare cases when the node is moving at a steady speed or when it randomly changes direction [55, 56].

The Markov Chain approach to modeling wireless communication errors has been very useful in evaluating the performance of many network protocols that were ini-

tially designed for wired networking. This can be used to measure their adaptability to wireless communications [44].

3.4 Consensus Theory Overview

The main focus of this study is going to be directed discrete-time wireless sensor networks in order to make the outcomes of this study more relevant to wireless sensor networks. Due to communication errors introduced by different factors such as hidden or exposed terminal problems in WSNs, full-duplex communication between any two given nodes within communication range of each other cannot be always guaranteed. As a result, directed graphs better model the network nodes and links between them and will be assumed necessary for any consensus algorithm proposed.

Most studies that approached the average consensus problem assumed a variation of the following problem framework:

Given a network of N "decision-making" agents, modeled by the 2-tuple dynamic graph $G(V, E(t))$, where each agent or node $i \in V$ has some initial reading or *state-value* $x_i(0)$, the goal of an average consensus algorithm is to have all agents in the network converge to the following *agreement space* [22]:

$$x_1 = x_2 = x_3 = ... = x_N \tag{3.1}$$

When the average consensus algorithm halts, each state-value in the network is going to be equal to the average of all the initial state-values:

$$\bar{x} = \sum_{\imath \in V} x_i(0) \tag{3.2}$$

The average consensus algorithm assumes that each agent i is capable of exchanging its state-value with its neighborhood $\mathcal{N}_i$ after each iteration of the algorithm.

The discrete-time form of the iterative average consensus algorithm is formulated as follows:

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j \in \mathcal{N}_i} a_{ij}(x_j(k) - x_i(k)) \tag{3.3}$$

The same algorithm can be expressed in matrix-multiplication form as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{P}\boldsymbol{x}_k \tag{3.4}$$

The Perron matrix [57] is defined as follows:

$$\boldsymbol{P} = \boldsymbol{I} - \epsilon \boldsymbol{L} \tag{3.5}$$

where $\mathbf{I}$ is defined as the identity matrix, ϵ is a step size and $\mathbf{L}$ is a Laplacian matrix defined as:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{3.6}$$

$\boldsymbol{D}$ is defined as the degree matrix of the graph representing the network and $\boldsymbol{A}$ is its adjacency matrix.

The idea of using the Perron matrix stems from discrete-time models such as the Markov Chain which was also embraced by other studies such as [22, 58, 59].

An N-state Markov Chain is defined as follows:

$$\boldsymbol{\pi(k+1) = \pi(k)P} \tag{3.7}$$

The row vector $\pi(k)$ is the distribution of the states of the Markov Chain and $\boldsymbol{P}$ in this case is the transition probability matrix such that the entry $P(i, j)$ is the transition probability from state i to state j. This matrix is non-negative and stochastic (the row sums add up to 1).

If the given Markov Chain is irreducible (the associated graph is strongly connected) and ergodic (there exists one maximum eigen-value associated with the corresponding transition matrix), the following applies when the distribution vector is multiplied by $\boldsymbol{P}$ over time:

$$\lim_{t \to \infty} \boldsymbol{\pi_t} = \boldsymbol{\pi^*} \tag{3.8}$$

The vector $\boldsymbol{\pi}^*$ is the stationary distribution of the Markov Chain [60].

Likewise, if the Perron matrix is non-negative and stochastic, consensus will be achieved by the corresponding network. Furthermore, if the Perron matrix is doubly-stochastic, where both row and column sums are equal to 1, average consensus will be reached because the associated directed graph will be balanced in this case [22,61,62].

Perron [57] showed that the Algorithm 3.5, given a network modeled by a strongly connected graph, will asymptotically lead to convergence. However, if the directed graph representing the network turns out to be balanced where the in-degree of each node is equal to its out-degree and the Perron matrix is doubly stochastic, average consensus will be asymptotically reached. Furthermore, the speed of convergence is proven to be equal to the second smallest eigen-value of the Laplacian matrix $\lambda_2(L)$ associated with the network.

This convergence rate analysis is more relevant to static networks, where no edges are activated or de-activated over time. It is a much more difficult problem to apply this analysis to switching or dynamic networks where the corresponding adjacency matrix of the network is time-varying resulting in a dynamic Laplacian matrix for the given network. The main difficulty is maintaining the spectral properties of the network stated above that allows it to converge to average consensus. This is discussed by many different studies including [15, 17, 63, 64].

While Perron [57] outlines the theoretical framework for achieving average consensus, it does not take into consideration the problems that usually arise in wireless communications such as communication errors, hidden terminal problem, etc. Also, this theory assumes that each node in the network stores the weight matrix associated with iterative consensus algorithm, even the ones that it never communicated with, not to mention that it does not handle the changes in the weight matrix when the power of a node dies out. Moreover, the rate of convergence to consensus represented by the second smallest eigen-value of the Perron matrix is not bounded. The assumptions of this theory are difficult to meet in real-world networks where there is

no guarantee that the directed network is always going to be balanced, which is the necessary condition to achieve average consensus.

Pappas et. al. [59] proposed to associate every node in an undirected network G with the stochastic matrix of a Markov chain. A unique stochastic matrix $\boldsymbol{S_i}$ is to be constructed by every node i based on a certain sparsity pattern $\boldsymbol{P}$ associated with the network G by sampling a set of 0-1 stochastic matrices that form the vertices of the convex polyhedron of the stochastic matrix to be constructed. Using the convergence properties of the Markov processes, the algorithm will converge to a consensus matrix $\dot{\boldsymbol{S}}$ using the following algorithm:

$$\dot{\boldsymbol{S}}_{\boldsymbol{i}} = -\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}(\boldsymbol{S_i} - \boldsymbol{S_j}), \qquad \forall i = 1, \cdots, m, \tag{3.9}$$

The convergence rate to consensus is equal to the minimum second smallest eigenvalue of the Laplacian matrix associated with the network G. Once $\dot{\boldsymbol{S}}$ is reached, the equilibrium distribution of the initial states of the network can be computed.

This study relies on each node constructing a stochastic matrix based on a certain sparsity pattern $\boldsymbol{P}$ of the network, and then all the nodes exchange their respective stochastic matrices with each other in order to compute the consensus stochastic matrix $\dot{\boldsymbol{S}}$. First, pre-defining a sparsity pattern for the network assumes that the initial position of each sensor node in the network is going to be pre-defined, which makes this proposal unsuitable for WSNs where the nodes are often randomly scattered in the topology where they are deployed. Second, the process of sampling the 0-1 stochastic matrices in each node to construct stochastic matrices and the process of transmitting and receiving those stochastic matrices and running the computationally-intensive algorithm 3.9 is unsuitable for modest computational and storage capabilities of the nodes in WSNs. Third, there is no bound established on the convergence rate to the stochastic matrix $\dot{\boldsymbol{S}}$.

Another scheme for consensus was proposed by Yin et. al. [58] which assumed that node state-values are communicated over a stochastically switching network represented by the strongly connected graph G. Also, they proposed a stochastic ap-

proximation algorithm that adjusts the weights on the network links as they claimed that fixed weighs will never achieve average consensus due to dynamic nature of the switching network.

Any given node in the network is expected to construct a Markov chain $\boldsymbol{X^i(k)}$ as:

$$\boldsymbol{X^i(k)} = e_j \quad \Rightarrow \quad (j,i) \in \mathcal{E}(k) \tag{3.10}$$

which is based on the sequence of signals it receives from its neighborhood $\mathcal{N}_i$. This study employs the well known distributed averaging algorithm:

$$\boldsymbol{s^i(k+1)} = \boldsymbol{s^i(k)} + \sum_{j=1}^{n} \boldsymbol{W}_{ij}(k+1)\mathcal{I}_{ij}(\boldsymbol{s^j(k)} - \boldsymbol{s^i(k)}) \tag{3.11}$$

but adds to it a binary indicator function $\mathcal{I}$ that will be employed when a a transmission fails at a certain iteration.

Using several constant-weight schemes, Yin et.al. demonstrated that fixed weights in the stochastic network settings assumed in their study would most likely lead to an unpredictable arbitrary consensus. Hence, the following adaptive weighting scheme was adopted:

$$\boldsymbol{W}_{ij}(k) = \frac{\alpha}{\hat{\pi}^{ij}(k)}, \quad \alpha > 0, \tag{3.12}$$

which the authors claim can improve the convergence to consensus by allowing each node to predict when its respective Markov chain is going to switch state and receive transmission from a different neighbor.

Yin et.al. [58] modified the distributed averaging algorithm to incorporate transmission failures which is a very relevant problem that often arises in WSNs that can prevent the network from reaching average consensus. However, it fails to incorporate any stochastic model for the transmission failure into the consensus algorithm; instead it employs a weighting scheme that helps each node predict transmission failure without taking any measures to reduce the effect of this failure on reaching average consensus. In addition, the convergence figures used to illustrate the theoretical results indicate that several thousand time steps are required for the initial state

values to converge to consensus. This is not suitable for WSNs due to the limited power and computational resources available in each node.

Boyd et.al. [21] considered the problem of each node in a given network G estimate the value of the average consensus without the knowledge of the network topology and using fast linear iterations. The study focused on devising a simple way to compute average consensus without incurring much overhead in terms of data communication and memory storage. Other studies such as [65–69] devised more sophisticated ways to estimate the average consensus value.

The following distributed average consensus algorithm was adopted [20]:

$$\boldsymbol{x}_{k+1} = \boldsymbol{W}\boldsymbol{x}_k \tag{3.13}$$

The Average Consensus algorithm can be reformulated as follows [23]:

$$\bar{\boldsymbol{x}}_{k+1} = \boldsymbol{W}^t\boldsymbol{x}(\boldsymbol{0}) \tag{3.14}$$

The previous equation shows that a discrete-time Markov Chain model can be used to model the problem of computing the average consensus where weight matrix W would be the transition probability matrix and the the average consensus vector the stationary distribution.

The goal of Boyd's study [20] is to minimize the convergence time to average consensus. Hence, the measure of the rate of convergence to consensus was defined as the spectral radius of the weight matrix.

Furthermore, the Metropolis weight matrix was defined:

$$W_{ij}(t) = \begin{cases} \frac{1}{1+\max\{d_i(t),d_j(t)\}} & \text{if}(i,j) \in E(t), \\ 1 - \sum_{\{i,k\}\in\mathcal{E}(t)} W_{ik}(t) & \text{if} \quad i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.15}$$

This weighting scheme was derived from methods of constructing Markov chains on graphs [70]. The advantage of the Metropolis scheme is that each node needs to only know the in-degree and out-degree of each node in its neighborhood. There is

no need to accumulate information about the whole network topology thus saving significant amount of memory and processing power. Consequently, the Metropolis weighting scheme is very suitable to the distributed average consensus algorithm.

However, Boyd et. al. [21] did not incorporate any wireless communication error model to the Metropolis weighting scheme. Consequently, the impact of communication errors on the performance of their proposed average consensus algorithm was not clear.

Mesbahi et.al. [71] introduced the notion of modeling a switching network using a random graph, which is defined as a graph where an edge is formed between any two given vertices with a certain probability p. If this probability is set to zero for all edges, the resulting graph is the empty graph. On the other hand, if this probability is set to one for all edges, the resulting graph is the static graph. The value of the probability can be set to a constant or it can be a variable that is generated by some function. Thus, the expected number of edges formed in a graph can be modeled as a random variable that has a binomial distribution as each edge in the graph has a Bernoulli random variable associated with it. Hence, the graph itself can be modeled as a sum of all of those Bernoulli random variables.

Since the adjacency matrix of the graph is dynamic, the degree of each node in the graph can be modeled by a random variable defined as a sum of the Bernoulli random variables that represent the different edges. As a result, the Laplacian matrix representing the network will become dynamic and dependent on the edge probability [71].

Using random graphs makes proving convergence to consensus relatively easy. One of the main conditions that need to be met for convergence to occur is that the graph representing the network needs to be connected. This can be proven by showing that the union of all the possible random graphs will be connected [71].

The rate of convergence analysis was reformulated using the random graphs' theory presented. It was concluded that the rate of convergence in random graphs with fixed edge probabilities will improve the rate of convergence because fixing the prob-

abilities will improve the "robustness" of the random graph. This was demonstrated using simulations for different numbers of nodes.

While Mesbahi et.al. [71] showed that using the notion of random graphs simplifies proving convergence to consensus and analysis of the rate of convergence, it did not clarify the reasons why a random graph can model a switching network.

Failure of communication links in a switching network is usually modeled by a wireless communication error model, like the fading signal model. Such models take into account many factors such as communication protocols, distance between sender and receiver, density of nodes in the topology considered, presence of interference that might be relevant in certain applications of WSNs, etc. Assuming that all such factors are simply random does not properly model communication channels in WSNs.

Furthermore, the conclusion of the study [71] that fixing the edge probabilities will improve the robustness of the graph was not relevant to WSNs. This is because a portion of the nodes in a WSN is expected to run out of power and no longer function over time, which means that the probability of forming an edge with such nodes will be zero. Fixing the probability of forming edges does not account for the power constraints of the WSN or any of the other factors that might cause the nodes to quit working. Thus, improving the robustness of the graph described [71] is not realistic in the WSN.

Kar et. al. [72] studied the problem of achieving average consensus in WSN where the communication links fail randomly. Thus, they assumed that the edge set of the graph representing the WSN is dynamic because they assumed that each link in the WSN can fail independently of other links in the network. Furthermore, they defined the matrix of edge formation probabilities for each possible link in the network:

$$P(i,j) = \begin{cases} p(i,j) & \text{if}(i,j) \in E(t), \\ 0 & \text{otherwise.} \end{cases} \tag{3.16}$$

As a result, the edge set *E(t)* was defined as a subset of $\mathcal{E}(t)$, the set of all edges that can possibly exist in the given WSN.

The average consensus algorithm used was stated in matrix form:

$$\boldsymbol{x(i+1) = W(i)x(i)} \tag{3.17}$$

The matrix $\boldsymbol{W(i)}$ is the weight matrix at time i such that it was heuristically defined as follows [21]:

$$\boldsymbol{W(i) = I - \alpha L(i)} \tag{3.18}$$

This study used a simple weighting scheme where each edge formed is assigned the constant weight α at each iteration, claiming that this implementation eliminates a certain level of complexity associated with calculating the weight of each link activated at time i.

Kar et. al. [72] concluded that the weight matrix $\boldsymbol{W(i)}$ is random because the Laplacian matrix of the network is random as the network links fail randomly as well. As a result, the convergence properties of the average consensus algorithm need to be analyzed from a probabilistic point of view.

Consequently, Kar [72] decided to study the convergence of the following mean square process in the standard Euclidean form as the necessary condition to achieve average consensus:

$$\lim_{i \to \infty} E[\| \, x(i) - \bar{x} \, \|_2] = 0 \tag{3.19}$$

The spectral radius of the weight matrix $\boldsymbol{W(i)}$ is still needed to be less than one in absolute value for convergence to take place.

Furthermore, Kar et. al. [72] defined the expected Laplacian matrix of the WSN as:

$$\boldsymbol{\bar{L} = E[L]} \tag{3.20}$$

Furthermore, the algebraic connectivity of the expected Laplacian matrix $\boldsymbol{\bar{L}}$ needed to be less than one for convergence to take place.

The authors [72] added another definition for the expected Laplacian:

$$\bar{\boldsymbol{L}} = \bar{\boldsymbol{D}} - \bar{\boldsymbol{A}} \tag{3.21}$$

If $\bar{\boldsymbol{A}}$ is irreducible, which means that the associated graph is connected, the algebraic connectivity of the Laplacian matrix is greater than zero.

Given the following spectral graph theory result [73]:

$$\lambda_N(L) \leq 2d_{max}(G) \tag{3.22}$$

The constant $2d_{max}(G)$ is defined as the maximum vertex degree of the given graph. Kar et.al. [72] claimed that the average consensus algorithm will converge in the mean-sense if the each edge weight is defined as:

$$\alpha_{ms} = \frac{1}{2d_{max}} \tag{3.23}$$

In order to maximize the convergence rate to average consensus, the authors [72] claimed the value of α_{ms} must be chosen in such a way that minimizes the algebraic connectivity of the expected Laplacian matrix.

The study of [72] provided a realistic model of a switching network and seemed to have provided a simple solution to the weighting problem of the average consensus algorithm. However, the assumption of having the link failures to be random was not clearly justified or connected with any communication errors relevant to WSNs. Furthermore, the proposed weighting scheme cannot be easily applied in a deployed WSN because computing minimum α_{ms} is too complex to be done in a decentralized manner. This is because each node needs to know the maximum vertex degree in the whole network, which is dynamic in itself as nodes might die overtime causing the degrees of the remaining ones to decrease. The authors assume that each node in the network is going to have the edge weight value pre-programmed which does not leave much room for updating the edge weight if a significant part of the network is lost upon deployment. This also means that node will have to be pre-positioned, which undermines one of the most important features of the WSN.

The ideas presented in the study of Kar [72] were further extended in [74–76] to include noise introduced to the communications between the nodes of the WSN. Kar et.al. [75] proposed to incorporate both noise and random link failures into the iterative average consensus algorithm. This is relevant when a certain transmission between any two given nodes succeeds, but the transmission itself is corrupted due to noise introduced by the communication channel established.The various studies discussed considered link failure only, others considered incorporating topology-related factors into the link failures [77].

Another study of Kar [75] proposed to model the changes in topology that occur over time by making the degree of each node in the network time varying, which was also assumed for weights of the communication links. However, the weights for all links in the network are going to have the same value at each time step. Furthermore, the effect of quantization error is also taken into account.

It is explained by another study [74] that when a transmission actually succeeds to make it from a sender to a receiver, some error is going to be introduced to the transmission due to channel noise, quantization, etc. This error could possible lead to divergence because the transmission itself no longer represents the state-value of the sender node. Thus, the value of the transmission itself need to be redefined so that it will account for the error introduced.

The impact of the random link failures were modeled by Kar [74] as "a sequence of independent identically distributed Laplacian matrices with mean defined as $\bar{L} = E[L(i)]$". Thus, during a particular iteration of the average consensus algorithm, a link may fail independently of the other links in the network. A noise sequence was defined to be added to the state-values of the nodes.

Formal proofs were formulated to show that convergence to consensus can actually take place with very high probability. Also, it was proven that convergence to will take place towards a finite random variable θ, which can be defined as an estimate of the desired average consensus value [74].

Furthermore, it was shown that by introducing a certain scheme to update the weights of the communication links, it would be possible to minimize the mean square error of θ. However, this was proven to affect the rate of convergence to consensus, which is usually not desired when running the average consensus algorithm in the WSN [74].

The study of [74,75] introduced another level of complexity to the average consensus problem, taking into account the channel noise and quantization errors. However, it was not clarified how much this error introduced is going to affect the convergence to average consensus.

Channel noise is often measured in terms of power. Thus, low power noise is not likely to affect the convergence to average consensus. Hence, it was important to quantify the impact of introducing the noise to the transmissions and distinguish between noise that will affect the convergence to average consensus and other that does not.

Also, the introduction of high levels of noise from the communication channel can be assumed to be a transmission failure. Thus, it could be assumed that the communication error model can be upgraded to take the channel noise and quantization into consideration, such that the link between two nodes would be considered to be broken if the channel noise level exceeds a certain predefined power threshold.

Moreover, the impact of introducing the channel noise on the rate of convergence was not clarified. Since WSNs usually have a limited power resource, designers of the any WSN protocol or algorithm need to be concerned with minimizing communication of data. Thus, it is important to describe the impact of channel noise on the rate of convergence in order to predict the life-time of the WSN if it employs the proposed average consensus algorithm.

Bamieh et. al. [78] proposed a novel method of evaluating the performance of the average consensus algorithm. The assumptions that were made in this study about the switching network are similar to other studies where each link is assumed to have an independent failure probability. Furthermore, each node in the WSN was

assumed to transmit its weighted state-value to its neighborhood after each iteration of the average consensus algorithm. The study proposed to run the average consensus algorithm on the whole static network, then introduced a new term to the average consensus algorithm that will "undo" the effect of the error by subtracting the state-values associated with the links that failed.

Hence, a vector $\boldsymbol{B(i,j)}$ was defined were the value of entry i is 1, entry j is -1 and all other entries are set to zero. A Bernoulli random variable $\delta_{i,j}(k)$ was defined as follows:

$$\delta_{i,j}(k) = \begin{cases} 1 & \text{edge failed with probability} p(i,j), \\ 0 & \text{edge did not fail with probability} 1 - p(i,j). \end{cases} \tag{3.24}$$

The average consensus algorithm was reconstructed incorporating the Bernoulli random variable:

$$\boldsymbol{x(k+1)} = (\boldsymbol{A} + \sum_{\{i,k\} \in E} \delta_{i,j}(k) \boldsymbol{B_{ij}}) \boldsymbol{x(k)} \tag{3.25}$$

The rate of convergence to average consensus was measured in terms of the deviation of the state-values of the network from the the average consensus value as the topology of the WSN is continually changing due to communication link failures and hence the corresponding Laplacian matrix is dynamic. Thus, it was proposed that using the traditional method of computing the rate of convergence to average consensus using the algebraic connectivity of the Laplacian matrix is not accurate and that measuring the rate of convergence in terms of the deviation from average consensus is more accurate.

The observation of Bamieh et. al. regarding the use of the algebraic connectivity of the Laplacian matrix of a switching WSN is very accurate. However, this study does not address any WSN specific problem or introduce any solution that can be implemented in a WSN. This is because the method of using the deviation from average consensus to measure the rate of convergence is very theoretical as it assumes that the value of average consensus is known in advance. Also, the idea that was proposed

to eliminate the effect of the failed links by introducing the Bernoulli random variable does not map to any real-world communication error model that could possibly occur in a WSN. This makes the whole convergence rate analysis introduced unrealistic as the error that is causing the network to become dynamic is poorly modeled.

Lopez et.al. [79] yet proposed another way to ensure convergence to average consensus in the presence of communication failures. The study focussed on tuning the value of the step-size α dynamically in order to account for noise that will affect the communication channels' robustness. The step-size itself is factored in the heuristic definition of the weight matrix of the network [20]:

$$\boldsymbol{W}(\boldsymbol{i}) = \boldsymbol{I} - \alpha \boldsymbol{L}(\boldsymbol{i}) \tag{3.26}$$

It was proposed by Lopez [79] that the two factors affecting a good estimate of a solution for the convergence problem is the noise power and the value of α.

According to Lopez [79], the proposed algorithm that is going to devise a sequence for the value of the step-size will have to make a trade-off between the rate of convergence to consensus and the accuracy of the mean square estimate of the average consensus value. Thus, a greedy approached was chosen to generate the sequence of step-sizes that will lead to average consensus.

However, this approach will require the pre-knowledge of the topology of the network considered. This problem was then resolved by assuming that the Laplcian of the network is always going to be circular, which means that each node in the network is going to have a fixed number of neighbors and hence a fixed degree matrix will be obtained for the whole network.

Moreover, if the nodes of the network considered are going to be randomly scattered in the given topology, then the idea of using a fixed degree value for each node will make sense. This is because each node is going to connect to other nodes that are within its communication range. The expected degree of each node can then be computed based on the "spatial distribution" of the nodes [79].

It was demonstrated by Lopez et.al. [79] that generating a step-size sequence can improve the performance of the iterative consensus algorithm. However, it implies from the heuristic weight equation that changing the step size will also change the value of the weights in the graph. It was not clear how this change in weights is going to be coordinated among the different nodes of the network given the fact that the communication links are not reliable. Such a proposition will require the knowledge of the topology of the network by each node, which is not possible as mentioned in the study.

Furthermore, the assumption that the degree of each node is totally reliant on the scattering the nodes in the topology is not completely accurate. This is because two neighboring nodes may fail at communicating with each other due to interference or other forms or noise that were not taken into consideration when this assumption was formulated. No communication error model was proposed to justify the presence of the failure in communication links. Also, nodes that run out of power may cause a change in the degrees of other nodes in the network, which was not taken into consideration in this assumption as well.

4. DISTRIBUTED AVERAGE CONSENSUS SCHEME

4.1 Consensus in Wireless Sensor Networks

The design of any algorithm for WSN, such as the average consensus algorithm, is influenced by many important factors [1]. One of the most important factors is fault tolerance [80], [81], [82] because sensors in the network are prone to failure, physical damage and interference in the environments they are deployed in. However, the algorithm running in the network should be able to achieve its task even after the failure of some of the nodes of the network. As a result, a decentralized scheme for average consensus is inevitable because the probability of a node failure in high in WSNs.

Another important factor influencing the design of WSN algorithms is scalability. Given a certain application the WSN is expected to be deployed in, the number of nodes used in the network can vary between a few nodes to several hundred of them in an area less than 10m in diameter [83]. As a result, several studies proposed different ways to evaluate the node density in a WSN [84]. Others [85], [86], [87], have listed typical node densities for different applications such as habitat monitoring and machine diagnosis applications. When the node density is too low, the graph representing the WSN can possibly be disconnected which means that the consensus performance is going to be affected [22]. On the other hand, having a high node density could overload the network thus inhibiting the ability of nodes to exchange information with each other.

Moreover, the power consumption of each node in a WSN is a major constraint that needs to be taken into consideration when designing the decentralized average consensus algorithm [1], [88], [89]. The power consumption in a WSN node takes place in three different functionalities: data sensing, communication and processing.

The power consumption of the data sensing component of the node is application and environment dependent. More power is going to be consumed in applications that requires the node to sense the environment over longer periods of times in order to make one measurement. More power is going to be consumed in noisy environments as the measurement is going to be repeated several times before it is taken. Furthermore, the circuity used to receive and transmit data consumes a significant amount of energy depending on its design and the size of the packets it communicates. Larger packets will consume more power to communicate. However, using smaller packet sizes was proven to be less efficient than using large packet sizes as a significant amount of inefficiency is going to be introduced turning on and off the circuitry of the communication component of the node [86]. Also, the processing component of the node is going to be application dependent. In the case of average consensus, each node is going to be expected to perform a number of additions and multiplications of scalar numbers relative to the amount of transmissions it receives from its neighbors; the higher the node density, the more power is going to be consumed by the average consensus algorithm. Consequently, minimizing the number of iterations needed to compute the average consensus will reduce the overall power consumption.

Most of the previous studies presented earlier assume that each node in a WSN can obtain all the most updated state values of its neighbors all at the same time. However, this is not going to be the case in a real-world WSN because packets sent by different nodes that are located at varying distances from the receiver node will have different propagation times. Also, nodes can run out of power or get damaged which means that their neighbors will never receive any data from them again. Furthermore, many communication errors are introduced in a WSN due to noise from the environment, interference from other nodes and other common wireless communication problems such as the hidden terminal problem, exposed terminal problem, etc.

As a result, the proposed decentralized average consensus algorithm is going to run in epochs, where each epoch is defined as a fixed period of time during which each

node tries to receive as many transmissions as possible. This gives the neighboring nodes the time to transmit their new state values and potentially retransmit them if the transmission fails for any reason.

Moreover, the fading signal model [90] is going to be incorporated in the proposed decentralized average consensus algorithm to model the communication errors that take place in a WSN. It is going to be used to compute the probability of communication between any two neighboring nodes in a given epoch, which will redefine the neighborhood of each node.

4.2 Problem Formulation

The WSN considered in this study is modeled by the graph $G(V, E(t))$, where V represents the set of nodes of the WSN and $E(t)$ is the dynamic edge set of the network at epoch t. $E(t)$ is a subset of $\mathcal{E}$, which is the set of all edges that could possibly be formed between the different nodes in the network. Each edge is assumed to represent a communication link; where each link connecting node i to node j has an independent success probability $p(i, j)$. For each epoch, a random probability matrix $\boldsymbol{P(t)}$ is generated for all links that could possibly exist in the WSN:

$$P_{ij}(t) = \begin{cases} p(i,j) & \text{if}(i,j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

$E(t)$ is equal to $\mathcal{E}$ if and only if no communication errors take place in the network during a certain epoch t. If a communication link is not formed between node i and node j, node i will not be able to transmit data to node j. On the other hand, if the communication link does not fail, then it is considered to be *active* and hence node i can transmit data to node j.

The neighborhood $\mathcal{N}_i$ of a certain node i is the set of all nodes that are located within communication range R of node i. This means that node i can communicate its state value, node ID and the number of nodes within its neighborhood that it

can communicate with (out-degree) to node $j \in \mathcal{N}_i$ during epoch t if and only if the edge $(i, j) \in E(t)$. $\mathcal{N}_i$ is dynamic because the nodes of the WSN are prone to failure, damage, power depletion, etc, which will cause the topology to change.

Each node i is assumed to make an initial analog measurement at the beginning of epoch 0. This reading is defined as the initial state of the node i; $x_i(0)$. Using linear distributed iterations, the goal of the average consensus algorithm is to have the state of each node converge to $\bar{x}$, the average of all the initial states values of all nodes $i \in V$ [78]:

$$\bar{x} = \sum_{\imath \in V} x_i(0) \tag{4.2}$$

During each iteration of the consensus algorithm, node $i \in V$ will broadcast its current state to all its neighbors $\mathcal{N}_i$. The average consensus algorithm does not assume any information regarding link success probabilities. It will just use whatever transmissions that it receives as input in any epoch. Also, each node cannot assume that its transmission was received by any of its neighbors.

The convergence to the equilibrium $\bar{x}$ will take place in each node $i \in V$ if and only if the algebraic connectivity of the Laplacian matrix L of the network is less than 1 in magnitude [21].

The rate convergence to average consensus τ is computed using the following equation [21]:

$$\tau = \frac{1}{\log \frac{1}{\rho}} \tag{4.3}$$

The average consensus algorithm will halt when the difference between the state value of each node $i \in V$ and $\bar{x}$ is minimized. Thus, it is desired after a running the average consensus algorithm for a number of epochs to achieve the following:

$$\lim_{t \to \infty} x_i(t) - \bar{x} = 0 \tag{4.4}$$

4.3 Metropolis Weighting Scheme

The proposed decentralized average consensus algorithm needs a weighting algorithm that allows it to modify the weights of the communication links when the neighborhood of each node changes every epoch such that the sum of the weights of the incoming communication links into every node is equal to the sum of weights of the outgoing communication links. As a result, the directed graph representing the WSN will be balanced. Thus, the condition for converging to the average consensus is met as the resulting weight matrix is symmetric such that its eigen-values are real and strictly less than one in magnitude [22].

At the end of each epoch and before running any iteration of the consensus algorithm, each node i is going to calculate the weight on each communication link it established with any of its neighbors j. Then, it is going to count the number of transmissions it received within the current epoch and use the out-degree values that it received with the transmissions to compute the Metropolis weights:

$$W_{ij}(t) = \begin{cases} \frac{1}{1+\max\{d_i(t),d_j(t)\}} & \text{if}(i,j) \in E(t), \\ 1 - \sum_{\{i,k\}\in\mathcal{E}(t)} W_{ik}(t) & \text{if } \ i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

where *E(t)* is the edge set of node i formed with its neighbors at epoch t. It is determined at each time step t using the fading signal error model.

The proposed consensus algorithm in this study uses the following matrix format:

$$\boldsymbol{x}_{k+1} = \boldsymbol{W_k}\boldsymbol{x}_k \tag{4.6}$$

where $\mathbf{W_k}$ is the Metropolis weight matrix and x_k is the state values' vector at iteration k.

The Metropolis weighting scheme was designed using theory for constructing fastest mixing Markov chains for a graph [70]. The weight matrix it produces every epoch is going to be symmetric and doubly stochastic. If the graph representing the

WSN is connected, the following necessary and sufficient conditions for convergence to average consensus are met [21]:

$$\mathbf{1}^T W = \mathbf{1}^T, W\mathbf{1} = \mathbf{1}, \rho(W - \mathbf{1}\mathbf{1}^T/n) < 1 \tag{4.7}$$

4.4 Fading Signal Error Model

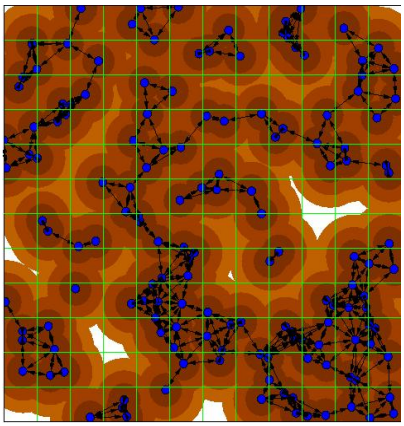

Fig. 4.1.: Node placement and communication range in the WSN topology

The reliability of wireless communication deteriorates in WSNs due to many factors such as interference, path loss and blockage. One of the most likely sources of wireless communication errors is fading, which is defined as the deterioration of signal power as it is transmitted from the sender to the receiver due to absorption of the signal power by the transmission medium or some obstacle lying between the sender and the receiver. The longer the distance between the sender and the receiver, the lower the signal to noise ratio. In order to accept a certain transmission, the receiver needs to detect a certain threshold power on the transmission; otherwise it will drop the transmission [91].

Consequently, the average consensus algorithm needs to have a fading signal error model incorporated into it in order to determine whether a particular link $(i, j) \in \mathcal{E}$ is going to be activated or not.

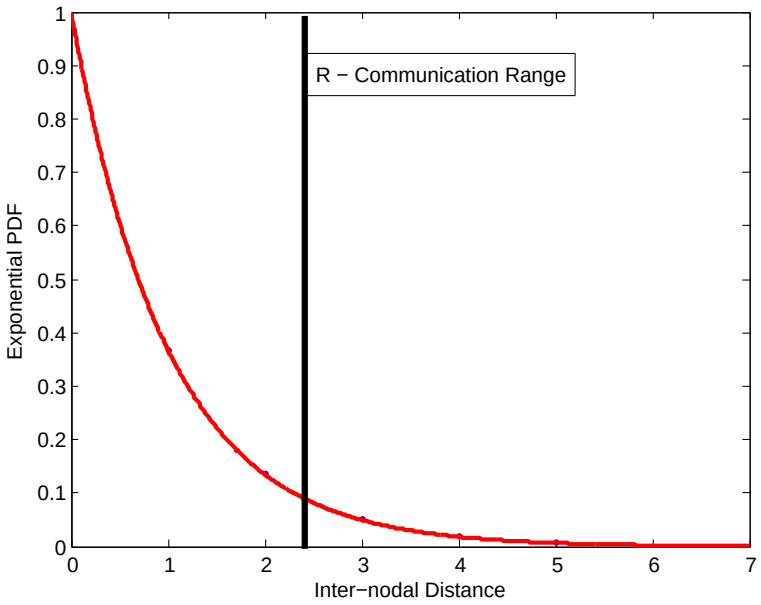

Fig. 4.2.: The truncated Exponential Probability Density Function (PDF)

A truncated form of the Exponential probability density function, shown in Fig. 4.2, can be used to model the fading effect on the signal between the sender and the receiver. Since the nodes are uniform-randomly scatted in the topology, the random variable X can model the distance between sender i and receiver j where both i and $j \in V$ [60]:

$$f_X(x) = \begin{cases} l + \lambda e^{-\lambda x} & 0 \leq x \leq R, \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

The parameter λ is defined in terms of the the communication range R of each node in the WSN and a constant A:

$$\lambda = \frac{A}{R} \tag{4.9}$$

The constant l is added to the truncated Exponential probability density function because it is only defined up to R such that any sender located outside the communication range R of the receiver would not be able transmit anything to the receiver.

Thus l offsets the area under the graph of the truncated Exponential probability density function so that it adds up to one:

$$f_X(x) = \begin{cases} \frac{e^{-A}}{R} + \frac{A}{R}e^{-\frac{Ax}{R}} & 0 \leq x \leq R, \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

The corresponding truncated Exponential probability distribution function for the fading signal model is:

$$F_X(x) = \begin{cases} 1 & x > R, \\ \frac{xe^{-A}}{R} - e^{-\frac{Ax}{R}} + 1 & 0 \leq x \leq R, \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

The truncated Exponential probability distribution function is then used to find the probability of forming a communication link between node i and node j using the inter-nodal distance as input. Then, this probability is compared to the corresponding random probability $p(i, j)$. The link will be established if the value generated by the fading signal model is smaller than the random probability associated with the link.

4.5 Approximation of the Fading Signal Error Model

The effectiveness of any proposed average consensus scheme is often measured in terms of the rate of convergence to average consensus. One of most complicated parameters in the proposed average consensus scheme to deal with in this analysis would be the inter-nodal distance between a node i and its neighbor $j \in \mathcal{N}_i$, introduced by the fading signal error model. This value is completely random in the range $[0, R]$ for each neighbor of i.

One way to eliminate the difficulty associated with the randomness of the inter-nodal distance is to approximate this value for all nodes in the WSN. It is assumed that the WSN nodes are uniform-randomly dispersed throughout the topology where

their WSN is going to be deployed. Hence, the distances between each node i and node $j \in \mathcal{N}_i$, will be random as well.

Let Y be a random variable that models the distance between node i and node $j \in \mathcal{N}_i$. At the end of each epoch, node i is going to broadcast its state-value to all its neighbors $\mathcal{N}_i$. Let $X_1, X_2, X_3, ..., X_N$ be a sequence of random variables that model observations of the distances between node i and any of its neighbors $j \in \mathcal{N}_i$. The goal is to obtain a good estimate of Y in terms of the observations $X_1, X_2, X_3, ..., X_N$.

Let $\hat{Y} = g(X_1, X_2, X_3, ..., X_N) = g(\vec{X})$ represent an estimate of Y. The error of the estimate is defined as:

$$\mathcal{E}(\vec{X}) = Y - \hat{Y} \tag{4.12}$$

The square of the error is defined as:

$$\mathcal{E}^2(\vec{X}) = (Y - \hat{Y})^2 \tag{4.13}$$

Since the error is a random variables as it is defined in terms of the of Y and the observations $X_1, X_2, X_3, ..., X_N$, $E[\mathcal{E}^2]$ represents the mean square error of the inter-nodal distance estimate. Hence, the goal is to find a good estimator that would minimize this mean square error.

Theorem 1: The best estimator of Y in terms of $X_1, X_2, X_3, ..., X_N$ is given by $\hat{Y} = E[Y|\vec{X}]$ [60].

Suppose $X_1, X_2, X_3, ..., X_N$ are independent and identically-distributed random variables (i.i.d.), then one possible estimator of $\hat{Y}$ can be defined as:

$$g(\vec{X}) = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{4.14}$$

Let the expected value of the random variables X_i be $E[X_i] = \mu$ and the variance $Var(X_i) = \sigma_i$. The expected value of the estimator is defined as:

$$E[\hat{Y}] = E[\frac{1}{n}\sum_{i=1}^{n} X_i] = \frac{1}{n}\sum_{i=1}^{n} \mu_i \tag{4.15}$$

The Law of Large Numbers [60]states that the average of the results obtained from performing the same independent experiment a large number of trials should be close to the expected value and will tend to become closer as more trials are performed.

The Strong Law of Large Numbers [60]states that for a given a sequence of independent random variables $X_1, X_2, X_3, ..., X_N$ with finite mean μ:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{\text{Almost Everywhere}} \mu \tag{4.16}$$

Hence:

$$P(\lim_{n\to\infty} \frac{X_1 + X_2 + X_3 + ... + X_N}{n} = \mu) = 1 \tag{4.17}$$

By the Strong Law of Large numbers, the expected value of the inter-nodal distance between the node $i \in V$ and its neighbors $\mathcal{N}_i$ can be used as an estimate of the distance distance between any node $i \in V$ and its neighbors $\mathcal{N}_i$.

These results are useful in formulating an approximation of the fading signal error model where the inter-nodal distance variable would be eliminated. This would reduce the complexity of analyzing the rate of convergence to consensus for the proposed average consensus algorithm.

Given the above truncated Exponential probability density function, the expected value of the distance X between the sender and the receiver is defined as:

$$E[X] = \frac{Re^{-A}}{2} - (\frac{R^2}{A} + \frac{R^2}{A^2})e^{-A} + \frac{R^2}{A^2} \tag{4.18}$$

In the approximated fading (AF) signal error model, the expected value of the inter-nodal distance is input to the fading model truncated Exponential probability distribution function to compute an *approximated fading signal threshold probability* γ, which is defined as the probability of turning a particular communication link (i, j) off given the defined fading signal model. On the other hand, $1 - \gamma$ will be the probability of activating the link (i, j).

Suppose that d is the expected inter-nodal distance between node i and node j where both nodes $i, j \in V$, the probability of activating the communication link (i, j), $1 - \gamma$, can be defined based on the fading signal error model probability distribution function as follows:

$$1 - \gamma = F_X(d) \tag{4.19}$$

If the previous formula is rearranged, γ can be defined as follows:

$$\gamma = e^{-\frac{Ad}{R}} - \frac{de^{-A}}{R} \tag{4.20}$$

4.6 Rate of Convergence to Consensus

In the communication link model established so far, the link (i, j) is established if the probability of the fading signal model is greater than the independent random probability $p(i, j)$ of the link. Consequently, γ less communication links are going to be established during any given epoch as a result of incorporating the approximated fading signal model. Also, the expected degree of each node $i \in \mathcal{N}_i$ is going to be decrease by γ. Thus, the expected degree matrix of the WSN is going to defined as:

$$\boldsymbol{D} = (1 - \gamma)\text{diag}(d_1, ..., d_N). \tag{4.21}$$

As a result, the expected Laplacian matrix L is going to be defined as follows:

$$L(i, j) = \begin{cases} (1 - \gamma) * \deg(V_i) & \text{if } \ i = j, \\ (1 - \gamma) * -1 & \text{if}(i, j) \in E(t), \\ 0 & \text{otherwise.} \end{cases} \tag{4.22}$$

The rate of convergence to average consensus is characterized by the second smallest eigen-value of the Laplacian matrix L. This rate is going to decrease by a constant factor γ as the number of edges formed in the graph decreased as a result of com-

munication failures [17]. Thus, it is expected that the convergence time to average consensus is going to increase by a constant factor γ.

5. SIMULATION

This section summarizes the simulation experiments that were carried out to evaluate the proposed scheme in the previous section, summarizes the outcomes of the different simulation scenarios and finally analyzes the outcomes in terms of their relevance to WSNs.

5.1 Setup

A simulation script was put together in order to evaluate the performance of the consensus scheme that was described in the previous section. The following table summarizes the parameters used in the simulations:

Table 5.1: Simulation Parameters

Parameter	**Value**
Topology Size	100 m $\times$ 100 m
Node Communication Range	10 m
Average Node Degree	3.99-17.56
Exponential Probability Distribution Function parameter - λ	0.5
Approximated Fading Signal Model Threshold probability - γ	0.1-0.25

The topology and the number of nodes were chosen to create node densities that would ensure that the graph representing the WSN will be strongly connected over time.

Extensive simulations were carried out with different topologies that have different average node degrees for both communication error models proposed earlier: the

fading model and the approximated fading error model. For the approximated fading error model, the same simulations were carried out for different γ values. Also, the same simulations were carried without any error model, which is the benchmark for evaluating consensus.

Several simulation scenarios were ran in order to collect the metrics that would evaluate the different aspects of the proposed consensus scheme.

5.2 Convergence to Average Consensus using proposed Scheme with No Error Model

The first set of simulations involved running the same simulation setup as described earlier without using an error model for different node numbers in order to verify that the proposed scheme converges to average consensus and to capture the number of iterations of the consensus algorithm needed for that purpose. Fig. 5.1 shows one sample of the simulations carried out for this purpose where the node number was set to 300. The average node degree for this scenario was approximately 10 and the average consensus value was approximately 50. As shown in Fig. 5.1, all 300 node state value converged to the average consensus value which is represented by the horizontal line on the graph. This needed less than 400 iterations of the consensus algorithm.

This simulation represents the convergence in a static, non-switching network, which was addressed in many previous studies such as [57]. It is obvious that the Metropolis weighting scheme was effective in enabling all the nodes in the network to converge to the average consensus value. This is going to be the benchmark simulation as it will later be compared to other simulations carried out that incorporated a communication error model.

5.3 Convergence to Average Consensus using proposed Scheme with the Fading Signal Error Model

In the second set of simulations, the fading error model was introduced to the scenario described earlier. Fig. 5.2 shows the impact of introducing the error model as almost 450 iterations were needed for the all the nodes to reach average consensus.

The increase in the number of iterations needed to converge to average consensus is a result of incorporating the fading signal communication error model. This is because introducing the model prevented forming many communication links such that each node has access to a smaller number of its neighboring state-values during each iteration. Since the algorithm employed computes the weighted average of neighboring state-values in order to estimate the average consensus value, more iterations were needed to accumulate enough state-values in order to achieve this purpose.

5.4 Convergence to Average Consensus using proposed Scheme with the Approximated Fading Error Model

Fig. 5.3 shows the impact of using the approximated fading error model for a relatively low value of γ set to 0.1. The number of iterations needed by the different nodes to reach average consensus is smaller than that for the fading model shown in Fig. 5.2. This value of γ did not cause many communication links to be broken. Thus, each node still had access to most of its neighboring state-values, if not all of them, at each iteration. This resulted in needing a similar number of iterations to the static network in order to converge to average consensus.

On the other hand, increasing the value of γ to 0.2 caused the number of iterations needed to reach average consensus to increase compared to the simulations where the value of γ was set to 0.1. This can be clearly observed in Fig. 5.4, which is similar to that of the fading model in Fig. 5.2. This is because this value of γ caused a larger number of communication links to fail for every node. As a result, each node

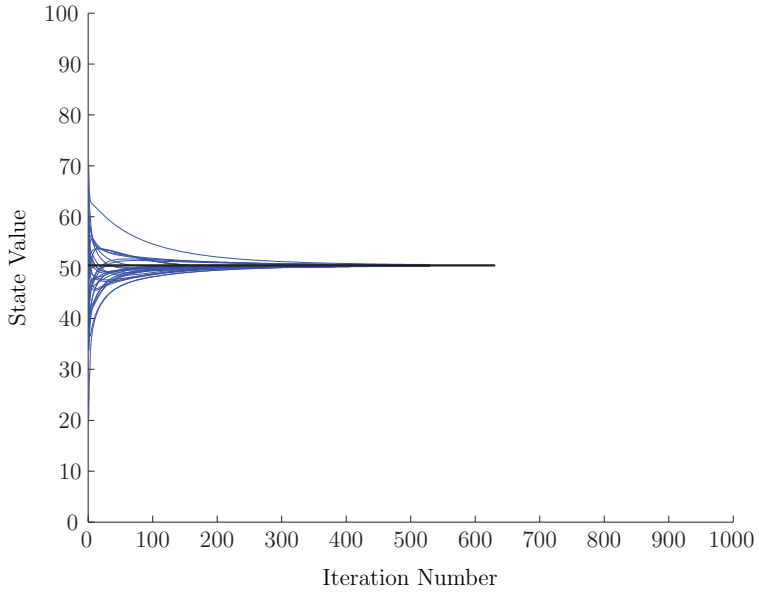

Fig. 5.1.: Static Network convergence to average consensus

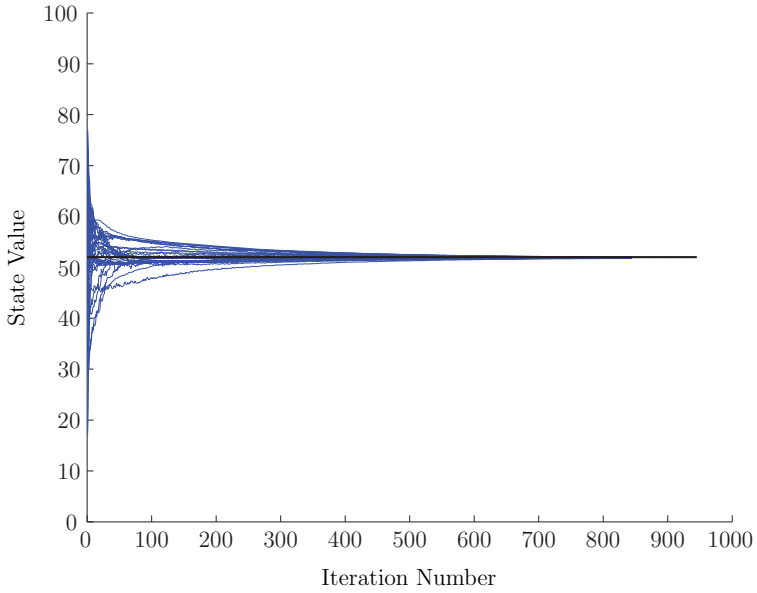

Fig. 5.2.: Fading Error Model convergence to average consensus

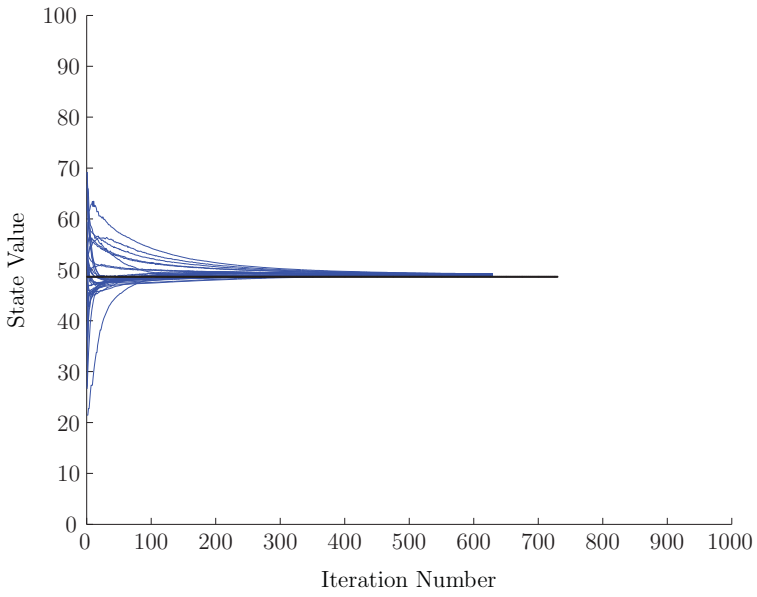

Fig. 5.3.: Approximated Fading Model convergence to average consensus - $\gamma = 0.1$

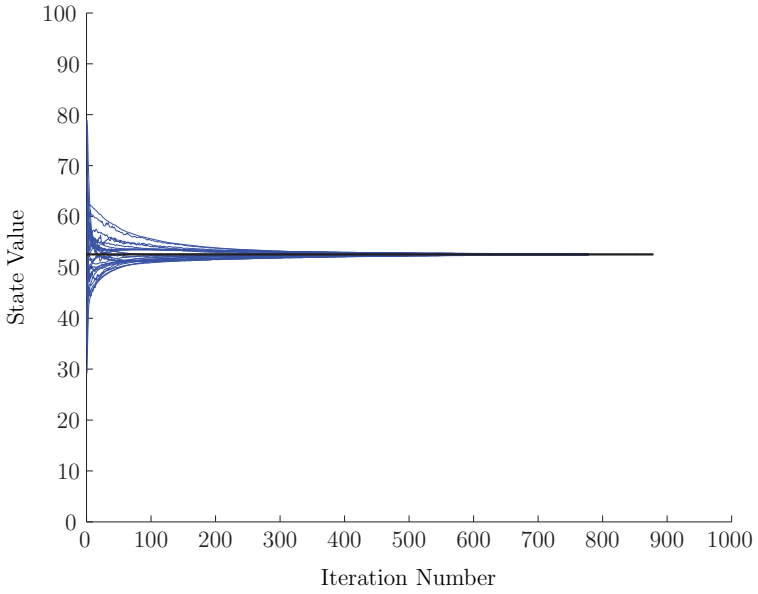

Fig. 5.4.: Approximated Fading Model convergence to average consensus - $\gamma = 0.2$

had access to a smaller number of neighboring state-values and hence needed a larger number of iterations to accumulate enough data to converge to average consensus.

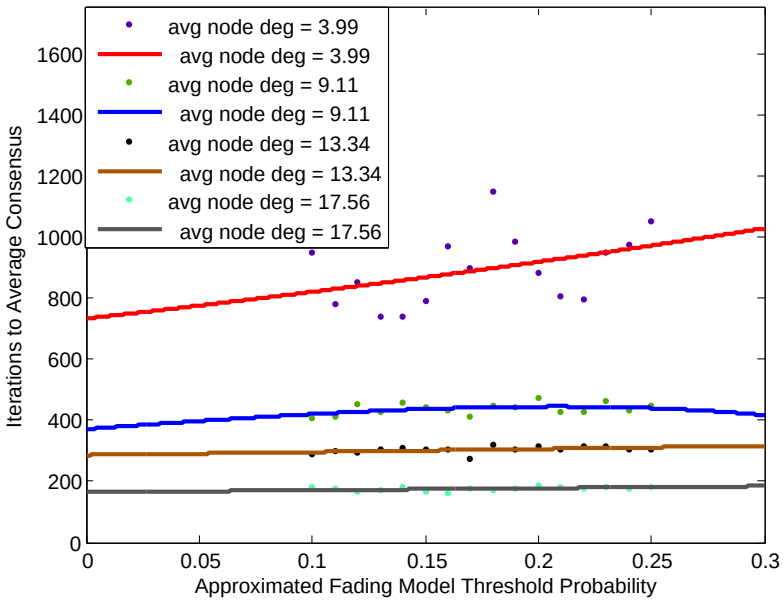

Fig. 5.5.: Iterations to average consensus vs. AF model threshold probability γ

Fig. 5.5 summarizes the effect of introducing γ on different topologies with varying average node degrees. Networks whose nodes have a smaller number of nodes on average, and hence lower average node degrees, needed more iterations to converge to average consensus than networks where nodes have larger numbers of neighbors. This is because introducing γ caused nodes with lower average node degrees to lose communication with a significant portion of its neighbors, which caused each node to have access to a smaller number of neighboring state-values. As a result, more iterations were needed to accumulate enough transmissions from the neighborhood in order to converge to average consensus. The larger the value of γ, the more communication links fail on average per node.

However, networks with larger average node degrees were not as affected, if at all, because each node has access to a large number of neighbors where losing a few of them won't much affect the number of state-values accumulated at each node. This will allow the network to converge to average consensus without needing more iterations of the consensus algorithms.

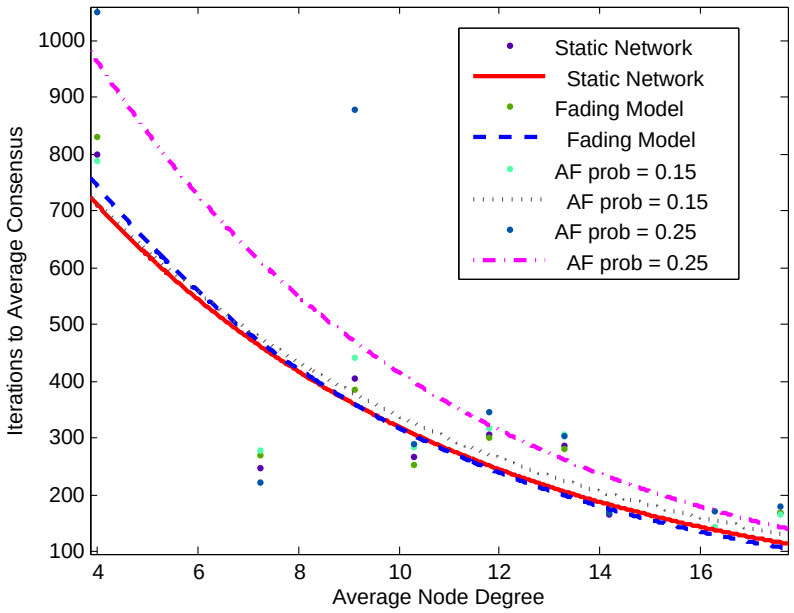

Fig. 5.6.: Iterations to average consensus vs. Average Node Degree

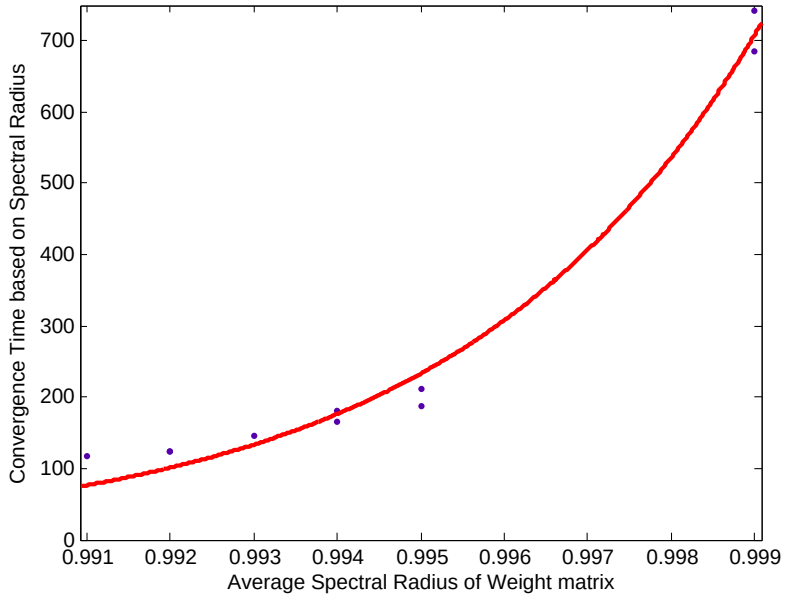

Fig. 5.7.: Convergence time vs. Spectral Radius

Fig. 5.6 summarizes the impact of using the different error models on the same simulation scenario. It is observed that using no error model needed the least number

of iterations to converge to average consensus, whereas the approximated fading model with γ set to 0.15 and the fading model simulations yielded close results. Given the value of λ specified in Table 5.1 as a parameter of the Exponential probability density function of the fading signal model, a γ value of approximately equal to 0.15 will be generated according to the analysis presented earlier. It is clear that behavior of the approximated fading model with $\gamma = 0.15$ is similar to the fading signal model itself.

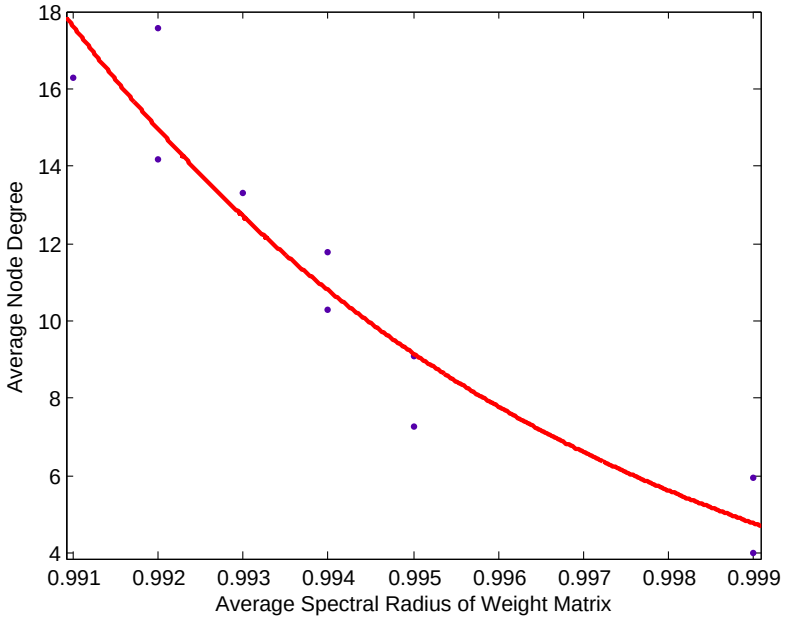

Fig. 5.8.: Spectral Radius vs. Average Node Degree

The scenario where the value of γ is set to 0.25 needed more iterations to average consensus than any other scenario presented. This shows that increasing the value of γ will increase the number of iterations needed to converge to consensus. However, networks with higher average node degrees will yield similar results no matter what error model is used. This is because the different error model did not reduce the number of communication links per node enough to make the convergence to average consensus need more iterations.

Fig. 5.9 shows the effect of increasing the value of γ on the convergence time to average consensus, which is computed using the the average spectral radius of the

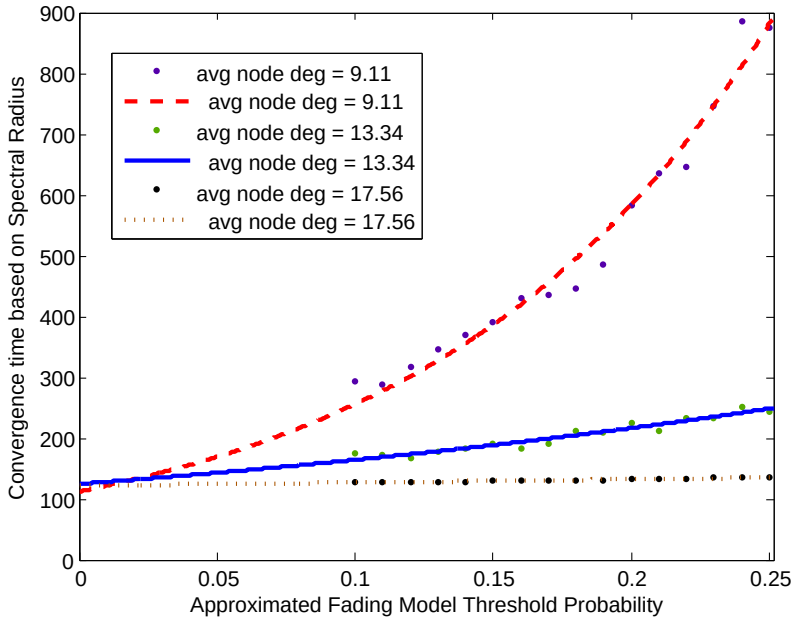

Fig. 5.9.: Convergence time vs. AF Model threshold probabilities γ

weight matrix of the WSN considered using the formula discussed earlier. Increasing the value of γ increased the convergence time to consensus more significantly for networks with lower average node degrees. On the other hand, networks with higher average node degrees were not much affected. This resembles the result that was obtained for iterations to consensus summarized in Fig. 5.5. Furthermore, Fig. 5.10 describes the impact of using different communication error models on the convergence time to average consensus. This result is similar to the one shown in Fig. 5.6 for iterations to consensus.

As mentioned before, the average spectral radius of the weight matrix of the WSN was used to compute the convergence time to average consensus. Fig. 5.7 shows that increasing the spectral radius value will increase the convergence time exponentially. Also, Fig. 5.8 shows that higher average node degree in the network results in a smaller average spectral radius of the WSN weight matrix. Hence, it is can be deduced from both figures that higher average node degree will need a smaller convergence time to average consensus, as expected. Moreover, Fig. 5.7 explains why the impact of

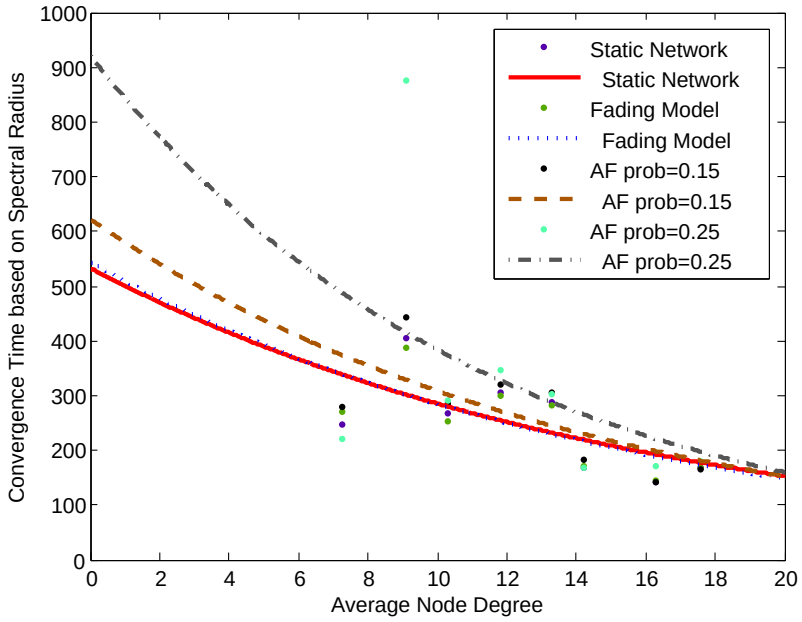

Fig. 5.10.: Convergence time vs. Average Node Degree

increasing the value of γ on the convergence time was sharper in Fig. 5.9 than its impact on iterations to consensus described in Fig. 5.5, even though the two results were still similar. This is because the convergence time tends to increase exponentially as the value of γ is increased while the iterations increase linearly in that respect.

Fig. 5.11 and Fig. 5.12 further explain the relationship between the convergence time to consensus and the number of iterations needed to converge to consensus. Fig. 5.11 shows the scenario were the average node degree is approximately 4. The iterations to consensus recorded seem to increase sharply then decrease slightly as the value of convergence time computed is increased. Fig. 5.12, which describes the scenario were the average node degree is approximately 17.5, shows that the relationship is linear.

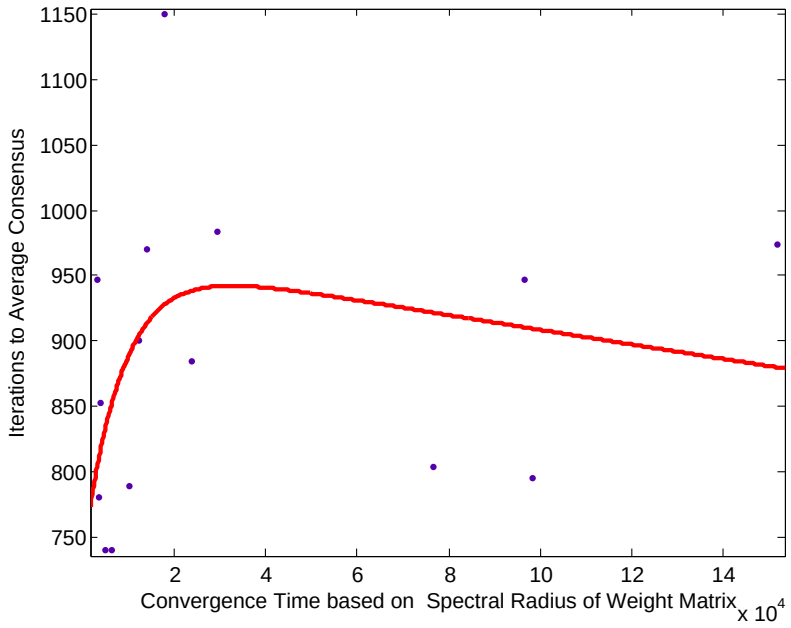

Fig. 5.11.: Iterations to Consensus vs. Convergence Time-avg node degree = 3.99

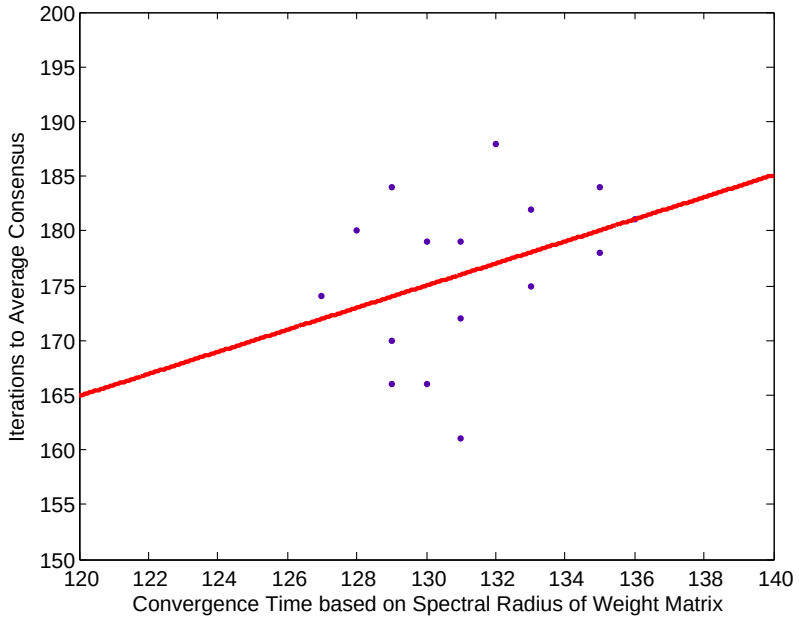

Fig. 5.12.: Iterations to Consensus vs. Convergence Time-avg node degree = 17.56

6. RESULTS ANALYSIS

This section will use the theory developed and simulations carried out earlier to highlight the different aspects of the proposed average consensus scheme and its relevance to WSNs.

6.1 The Significance of the Approximated Fading Signal Error Model

Introducing the fading signal error model to the proposed average consensus scheme was very relevant to WSNs. Furthermore, approximating this model was important for several reasons. First, it simplified the theoretical analysis of the proposed scheme by eliminating the need to consider the inter-nodal distance of each pair of neighboring nodes in the WSNs by taking advantage of the uniform random spatial distribution of the WSN nodes in the topology. Second, as each node in a WSN may not be able to measure the inter-nodal distance with its neighbors, it provides a means for it to predict whether a certain transmission is going to be delivered to a certain neighbor or not. If a given node predicts that the transmissions is not going to succeed, then it will defer it for later thus reducing the network overloading. Third, keeping track of delivered transmissions might give the node an idea about the rate to convergence and may allow it to predict when will the average consensus algorithm halt.

6.2 The Metropolis Weighting Scheme and the incorporated Communication Error Models

In all the simulations that were carried out, node state values converged to the average consensus value, which is the average of all initial state values of all nodes. This

is because the Metropolis weighting scheme allowed each node to generate weights for their different communication links for each iteration of the consensus algorithm that kept the overall weight matrix of the network doubly-stochastic over time, which is the property needed to converge to average consensus [57]. None of this required any node in the network to collect information about the topology or store any historic data. This enabled the average consensus algorithm to operate irrespective of topology changes induced by communication failures, power depletion in nodes, etc.

However, introducing the different error models increased the number of iterations needed by state node values to converge to average consensus. This is because each node will receive less state values in each epoch, and hence will not have enough data to quickly approximate the average consensus value using the consensus algorithm. Hence, using no error model while evaluating the consensus algorithm for WSNs is not realistic and gives misleading results about the performance of the algorithm in potential real world applications.

Introducing the fading model increased the number of iterations needed for convergence to average consensus the most since. This is because the generated probability of establishing a communication link between any two neighboring nodes is dependent on the distance between them, which results in assigning most communication links low success probabilities as a limited number of nodes can be very close to any given of their neighbors. The communication range of each node is already small as shown in Table 5.1.

Furthermore, Fig. 5.6 showed that the approximated fading signal error model with γ set to 0.15 has a similar impact as the fading signal error model, whereas using γ set to 0.1 had a similar impact as having no error model at all. This is because the using a smaller γ means that very few communication link will fail. As a result, the approximated fading signal error model with γ set to 0.15 can possibly be used to approximate the effect of the fading signal error model.

6.3 Rate of Convergence to Average Consensus

As shown in Fig. 5.9, increasing the value of γ increased the convergence time to average consensus. Earlier in this study, it was theoretically predicted that the increase in convergence time is going to be linear. This is clearly illustrated by the simulation of the WSN with average node degree equal to 13.34. However, simulations carried out with a lower number of nodes, and hence with a lower average node degree, have shown that the increase in convergence time was exponential. This is because networks simulated with a lower number of nodes had a lower algebraic connectivity as they had a smaller number of edges formed in the first place. When the error model was introduced, even more edges were deactivated and hence the impact on the convergence time was very significant, possibly disconnecting the network at times. However, average consensus was still achieved as the network does not need to be connected all the time to achieve average consensus [74].

On the other hand, simulations of networks with very large node number, and hence very large average node degree, did not seem to be affected at all by the introduction of the error model. This is because each node still received enough transmissions from its neighbors after the introduction of the error model to compute the average consensus value at the same rate.

Consequently, the theoretical analysis was accurate for simulations where the algebraic connectivity was moderate in the sense that the number of edges deactivated was increased by a linear factor only. In the case of lower algebraic connectivity, many disconnected components were created over time where the edges within those components were less useful in transmitting signals that contributed to computing average consensus, which was not taken into consideration by the theoretical analysis. Furthermore, in the case of the very high average node degree, the error model still did not break enough edges to make a difference in the convergence time, which was not reflected in the theoretical analysis as well. Increasing the value of γ in the network with very high average node degree will eventually affect convergence time,

but then it would be impossible to compare this scenario with networks with lower average node degrees as those will completely get disconnected and hence divergence will take place.

6.4 Rate of Convergence and Number of Iterations of the algorithm to Average Consensus

Fig. 5.9 shows that the number of iterations to consensus and the corresponding computed convergence time are linearly related. First, this illustrates that the rate of convergence to average consensus as measured by the average of the spectral radii of the weight matrices formed during the different epochs of the average consensus algorithm accurately measures the performance of the average consensus algorithm. Second, this relationship provides a way to predict the number of iterations the average consensus algorithm is going to run in a certain period of time given a network with a certain algebraic connectivity. Third, it shows that the algebraic characteristics of the graph representing a network can be used to estimate how many iterations the algorithm is going to run given a certain topology.

However, the convergence time does not take into consideration the propagation time of wireless signals and the processing times of the nodes. In a real-world scenario, this will definitely increase the rate of convergence to consensus.

6.5 Suitability of the proposed Scheme to WSNs

As discussed earlier, power is one of the most critical constraints in WSNs. Thus, any proposed WSN algorithm or protocol must take into consideration minimizing the power consumption of the WSN. Also, it was clarified earlier that the power of the WSN is mainly consumed in processing, communication and sensing, where communication is the most significant power consumer [1].

The simulations that were carried show that the number of iterations needed to reach average consensus is smaller than those needed by other proposed schemes such

as [58]. Moreover, each node only needs to communicate its state-value, ID, in-degree and out-degree, which is a relatively small amount of data. Also, the proposed scheme does not require any node in the WSN to run any lengthy or complex computations to calculate the average consensus value or the communication links' weights. This will significantly reduce the power consumption of each node.

Another issue that is usually taken into consideration in WSNs is the scalability of the network [1]. Since the Metropolis weighting scheme does not require any node in the WSN to store any information about the topology or about the other nodes, the proposed scheme should work the same way for both small and large networks. However, large WSNs deployed in a small topology will generate a larger node density. This could possibly lead to overloading the WSN as all nodes will be competing to broadcast their state-values concurrently.

7. CONCLUSIONS AND FUTURE WORK

7.1 Summary

In summary, a decentralized scheme to compute the average consensus of the initial values of all nodes in a WSN was introduced. Also, two different error model, the fading model and the approximated fading error model, were introduced to describe the probability of establishing communication links between the neighboring nodes. It was shown that the proposed scheme enabled all nodes to converge to average consensus.

However, the number of iterations needed to converge to consensus was greater when the error models were incorporated as compared to the error-free scenarios. Furthermore, it was shown that the fading model was the most realistic one in modeling the communication errors that usually occur in WSNs and that the approximated fading error model can possibly approximate the fading model.

The advantage of using the approximated fading error model is that it simplifies the mathematical analysis of the convergence to consensus as it eliminates the need to take the inter-nodal distance parameter into consideration. Also, from an implementation point of view, the approximated threshold probability γ that the approximated fading error model provides can help a given node in the network predict whether its transmission at a particular epoch is going to succeed or not. If the node determines that the transmission is going to fail, then it can defer the transmission for some time to reduce transmission collision and improve network throughput.

Moreover, the analysis of the simulations introduced has shown that increasing the value of γ is going to impact the number of iterations needed to converge to average consensus. This will not be as significant on networks with high average node degree as compared to others with lower average node degree.

The same results were obtained in terms of convergence time, which is calculated using the average spectral radius of the weight matrix of the WSN. However, the simulations that were carried out have shown that the relationship between the number of iterations to average consensus and convergence time based on spectral radius is linear for networks with high average node degree and almost logarithmic for networks with lower average node degree.

7.2 Future Work

There are many aspects of this study that still need to be explored. First, this study used the approximated fading signal error model to represent the communication failures in the WSN. There are more sophisticated methods to describe the behavior of the communication links using discrete-event systems such as the Markov Chain. Second, the proposed scheme should be assessed in terms of throughput, packet delay, power efficiency and other metrics that are often used to assess the performance of WSNs. Third, the average consensus scheme proposed may introduce some overhead that needs to be better studied. This will give a better estimate of convergence time to average consensus. Fourth, the proposed scheme should be upgraded to take power as a factor in the design of the consensus algorithm.

LIST OF REFERENCES

LIST OF REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] C. Perkins, *Ad Hoc Networks*. Addison-Wesley, 2000.

[3] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 56–67, ACM, 2000.

[4] M. Bhardwaj, T. Garnett, and A. Chandrakasan, "Upper bounds on the lifetime of sensor networks," in *ICC 2001: IEEE International Conference onCommunications, 2001.*, vol. 3, pp. 785–790, 2001.

[5] P. Bonnet, J. Gehrke, and P. Seshadri, "Querying the physical world," *IEEE Personal Communications,*, vol. 7, pp. 10–15, Oct 2000.

[6] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, (New York, NY, USA), pp. 174–185, ACM, 1999.

[7] C. Jaikaeo, C. Srisathapornphat, and C.-C. Shen, "Diagnosis of sensor networks," in *ICC 2001: IEEE International Conference on Communications, 2001.*, vol. 5, pp. 1627–1632, 2001.

[8] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor and position sensors," in *Conference On Microtechnologies in Medicine and Biology, 1st Annual International, 2000*, pp. 607–610, 2000.

[9] J. Rabaey, M. Ammer, J. da Silva, J.L., D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *Computer*, vol. 33, pp. 42–48, Jul 2000.

[10] B. Warneke, M. Last, B. Liebowitz, and K. Pister, "Smart dust: communicating with a cubic-millimeter computer," *Computer*, vol. 34, pp. 44–51, Jan 2001.

[11] S. Xi and X.-M. Li, "Study of the feasibility of VANET and its routing protocols," in *WiCOM '08: 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008.*, pp. 1–4, Oct 2008.

[12] F. Batool and S. Khan, "Traffic estimation and real time prediction using adhoc networks," in *Proceedings of the IEEE Symposium on Emerging Technologies,*, pp. 264–269, Sept 2005.

[13] S. Biswas, R. Tatchikou, and F. Dion, "Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety," *IEEE Communications Magazine,*, vol. 44, pp. 74 – 82, Jan 2006.

[14] F. Somda, H. Cormerais, and J. Buisson, "Intelligent transportation systems: a safe, robust and comfortable strategy for longitudinal monitoring," *IET Intelligent Transport Systems,*, vol. 3, pp. 188–197, Jun 2009.

[15] D. P. Bertsekas and J. N. Tsitsiklis, "Comments on coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control,*, vol. 52, pp. 968–969, May 2007.

[16] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, pp. 169–182, Feb 2005.

[17] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control,*, vol. 49, pp. 1520 – 1533, Sept 2004.

[18] M. Porfiri and D. Stilwell, "Consensus seeking over random weighted directed graphs," *IEEE Transactions on Automatic Control,*, vol. 52, pp. 1767–1773, Sept 2007.

[19] W. Ren and R. W. Beard, "Consensus of information under dynamically changing interaction topologies," in *Proceedings of the 2004 American Control Conference ACC*, 2004.

[20] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2003.

[21] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, (Piscataway, NJ, USA), p. 9, IEEE Press, 2005.

[22] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan 2007.

[23] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *J. Parallel Distrib. Comput.*, vol. 67, no. 1, pp. 33–46, 2007.

[24] F. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[25] S. Kar, S. Aldosari, and J. M. F. Moura, "Topology for distributed inference on graphs," *Available: http://www.arxiv. org/abs/cs/0606052*, 2006.

[26] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: mobile networking for "smart dust"," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, (New York, NY, USA), pp. 271–278, ACM, 1999.

[27] J. Rabaey, J. Ammer, J. da Silva, J.L., and D. Patel, "Picoradio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes," in *IEEE Computer Society Workshop on VLSI Proceedings, 2000.*, pp. 9–12, 2000.

[28] J. Rabaey, M. Ammer, J. da Silva, J.L., D. Patel, and S. Roundy, "Picoradio supports ad hoc ultra-low power wireless networking," *Computer*, vol. 33, pp. 42–48, jul 2000.

[29] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 166–179, ACM, 2001.

[30] A.-S. Porret, T. Melly, C. Enz, and E. Vittoz, "A low-power low-voltage transceiver architecture suitable for wireless distributed sensors network," in *ISCAS'00:The 2000 IEEE International Symposium on Circuits and Systems Proceedings, 2000.*, vol. 1, pp. 56–59, 2000.

[31] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, (New York, NY, USA), pp. 263–270, ACM, 1999.

[32] W. Chen, L. Chen, Z. Chen, and S. Tu, "Wits: A wireless sensor network for intelligent transportation system," in *IMSCCS'06:First International Multi-Symposiums on Computer and Computational Sciences, 2006.*, vol. 2, pp. 635–641, 20-24 2006.

[33] M. Franceschinis, L. Gioanola, M. Messere, R. Tomasi, M. Spirito, and P. Civera, "Wireless sensor networks for intelligent transportation systems," in *VTC'09: IEEE 69th Vehicular Technology Conference, 2009.*, pp. 1–5, 26-29 2009.

[34] B. Li, H. Wang, B. Yan, and C. Zhang, "The research of applying wireless sensor networks to intelligent transportation system(its) based on ieee 802.15.4," in *6th International Conference on ITS Telecommunications Proceedings, 2006*, pp. 939–942, june 2006.

[35] M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang, "Wireless sensor networks in intelligent transportation systems," *Wirel. Commun. Mob. Comput.*, vol. 9, no. 3, pp. 287–302, 2009.

[36] M. Zhang, J. Song, and Y. Zhang, "Three-tiered sensor networks architecture for traffic information monitoring and processing," in *IROS'05: IEEE/RSJ International Conference onIntelligent Robots and Systems, 2005.*, pp. 2291–2296, 2-6 2005.

[37] A. Skordylis, A. Guitton, and N. Trigoni, "Correlation-based data dissemination in traffic monitoring sensor networks," in *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, (New York, NY, USA), pp. 1–2, ACM, 2006.

[38] K. LA., *Sensor Technologies and Data Requirements for ITS*. Artech House, 2001.

[39] K. B., *Networking Wireless Sensors*. Cambridge University Press, 2005.

[40] S. Ergen and P. Varaiya, "Pedamacs: power efficient and delay aware medium access protocol for sensor networks," *IEEE Transactions on Mobile Computing,*, vol. 5, pp. 920 – 930, july 2006.

[41] Y. Wang, G. Zhou, and T. Li, "Design of a wireless sensor network for detecting occupancy of vehicle berth in car park," in *PDCAT'06: Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, 2006.*, pp. 115–118, dec. 2006.

[42] M. Karpiriski, A. Senart, and V. Cahill, "Sensor networks for smart roads," in *PerCom'06: Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2006.*, pp. 305–310, 13-17 2006.

[43] Y. Liu, A. Choudhary, J. Zhou, and A. Khokhar, "A scalable distributed stream mining system for highway traffic data," in *10th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, Sept. 2006.

[44] H. Bai and H. Aerospace, "Error modeling schemes for fading channels in wireless communications: A survey," *IEEE Communications Surveys and Tutorials*, vol. 5, pp. 2–9, 2003.

[45] B. Sklar, "Rayleigh fading channels in mobile digital communication systems .i. characterization," *IEEE Communications Magazine,*, vol. 35, pp. 90–100, jul 1997.

[46] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *In proceedings of the winter simulation conference*, pp. 597–604, 1996.

[47] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 243–254, ACM, 1996.

[48] T. S. Rappaport, *Wireless Communications*. Prentice Hall, 1996.

[49] J. Yee and J. Weldon, E.J., "Evaluation of the performance of error-correcting codes on a gilbert channel," in *ICC '94: IEEE International Conference on Communications, 1994.*, pp. 655–659, 1-5 1994.

[50] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A markov-based channel model algorithm for wireless networks," in *Wireless Networks*, pp. 189–199, 2001.

[51] H. S. Wang and N. Moayeri, "Finite-state markov channel-a useful model for radio communication channels," *IEEE Transactions on Vehicular Technology,*, vol. 44, pp. 163–171, feb 1995.

[52] Hwang, Seok, Kim, and Dongsoo, "Markov model of link connectivity in mobile ad hoc networks," *Telecommunication Systems*, vol. 34, pp. 51–58, February 2007.

[53] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *Int. J. Netw. Manag.*, vol. 11, no. 1, pp. 3–30, 2001.

[54] J. Tsumochi, K. Masayama, H. Uehara, and M. Yokoyama, "Impact of mobility metric on routing protocols for mobile ad hoc networks," in *PACRIM'03:IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2003.*, vol. 1, pp. 322–325, 28-30 2003.

[55] A. McDonald and T. Znati, "A mobility-based framework for adaptive clustering in wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications,*, vol. 17, pp. 1466–1487, aug 1999.

[56] M. Zonoozi and P. Dassanayake, "User mobility modeling and characterization of mobility patterns," *IEEE Journal on Selected Areas in Communications,*, vol. 15, pp. 1239–1252, sep 1997.

[57] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge Univ. Press, 1987.

[58] K. Topley, V. Krishnamurthy, and G. Yin, "Average-consensus with switched Markovian network links," in *12th International Conference on Information Fusion, 2009. FUSION '09.*, pp. 388–395, Jul 2009.

[59] M. Zavlanos, D. Koditschek, and G. Pappas, "A distributed dynamical scheme for fastest mixing Markov chains," in *American Control Conference, 2009. ACC '09.*, pp. 1436–1441, Jun 2009.

[60] S. U. P. Athanasios Papoulis, *Probability, Random Variables and Stochatic Processes*. McGraw Hill, 2001.

[61] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *in Proceedings of IEEE INFOCOM*, pp. 1653–1664, 2005.

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

[63] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control,*, vol. 50, pp. 169–182, Feb. 2005.

[64] W. Ren, R. W. Beard, S. Member, A. W. Beard, and S. Member, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," 2003.

[65] Z. Chair and P. Varshney, "Distributed bayesian hypothesis testing with distributed data fusion," *IEEE Transactions on Systems, Man and Cybernetics,*, vol. 18, pp. 695–699, sep/oct 1988.

[66] V. Delouille, R. Neelamani, and R. Baraniuk, "Abstract robust distributed estimation in sensor networks using the embedded polygons algorithm," in *in Proceedings of the third Internation Conference on Information Processing in Sensor Networks*, pp. 405–413, 2004.

[67] Z.-Q. Luo, "An isotropic universal decentralized estimation scheme for a bandwidth constrained ad hoc sensor network," *IEEE Journal on Selected Areas in Communications,*, vol. 23, pp. 735 – 744, april 2005.

[68] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *46th IEEE Conference on Decision and Control, 2007*, pp. 5492 –5498, 12–14 2007.

[69] J. N. Tsitsiklis, "Decentralized detection," in *In Advances in Statistical Signal Processing*, pp. 297–344, JAI Press, 1993.

[70] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM REVIEW*, vol. 46, pp. 667–689, 2003.

[71] Y. Hatano and M. Mesbahi, "Agreement over random networks," in *CDC:43rd IEEE Conference on Decision and Control, 2004.*, vol. 2, pp. 2010–2015, 14-17 2004.

[72] S. Kar and J. Moura, "Distributed average consensus in sensor networks with random link failures," in *ICASSP 2007: IEEE International Conference on Acoustics, Speech and Signal Processing, 2007.*, vol. 2, pp. II–1013 –II–1016, 2007.

[73] S.Camix and S.Stephani, *Matrices and Graphs: Theory and Applications*. World Scientific Publishing, 1996.

[74] S. Kar and J. Moura, "Distributed average consensus in sensor networks with random link failures and communication channel noise," in *ACSSC 2007:Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers, 2007.*, pp. 676–680, 4-7 2007.

[75] S. Kar and J. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Transactions on Signal Processing,*, vol. 57, pp. 355–369, jan. 2009.

[76] U. Khan, S. Kar, and J. Moura, "Diland: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Transactions on Signal Processing,*, vol. 58, pp. 1940–1947, march 2010.

[77] M. Porfiri and D. Stilwell, "Stochastic consensus over weighted directed networks," in *ACC'07:American Control Conference, 2007.*, pp. 1425–1430, 9-13 2007.

[78] S. Patterson, B. Bamieh, and A. El Abbadi, "Distributed average consensus with stochastic communication failures," in *46th IEEE Conference on Decision and Control, 2007.*, pp. 4215–4220, 12-14 2007.

[79] C. Mosquera, R. Lopez-Valcarce, and S. Jayaweera, "Stepsize sequence design for distributed average consensus," *IEEE Signal Processing Letters,*, vol. 17, pp. 169–172, feb. 2010.

[80] G. Hoblos, M. Staroswiecki, and A. Aitouche, "Optimal design of fault tolerant sensor networks," in *Proceedings of the 2000 IEEE International Conference on Control Applications, 2000.*, pp. 467–472, 2000.

[81] D. Nadig, S. Iyengar, and D. Jayasimha, "A new architecture for distributed sensor integration," in *Southeastcon '93,IEEE Proceedings,*, p. 8 p., apr 1993.

[82] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," *IEEE Personal Communications,*, vol. 8, pp. 52–59, aug 2001.

[83] S. Cho and A. Chandrakasan, "Energy efficient protocols for low duty cycle wireless microsensor networks," in *ICASSP '01: IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings, 2001.*, vol. 4, pp. 2041–2044, 2001.

[84] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks: Self-configuring localization systems," in *ISCTA 01: In proceedings of the sixth international symposium on communication theory and applications*, 2001.

[85] E. Petriu, N. Georganas, D. Petriu, D. Makrakis, and V. Groza, "Sensor-based information appliances," *IEEE Instrumentation Measurement Magazine,*, vol. 3, pp. 31 – 35, dec 2000.

[86] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 272–287, ACM, 2001.

[87] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *ACM Commun.*, vol. 47, no. 6, pp. 34–40, 2004.

[88] A. Giridhar and P. R. Kumar, "Maximizing the functional lifetime of sensor networks," in *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, (Piscataway, NJ, USA), p. 2, IEEE Press, 2005.

[89] J. C. Dagher, M. W. Marcellin, and M. A. Neifeld, "A theory for maximizing the lifetime of sensor networks," *IEEE Transactions on Communications,*, vol. 55, no. 2, pp. 323–332, 2007.

[90] H. Bai and M. Atiquzzaman, "Error modeling schemes for fading channels in wireless communications: A survey," *IEEE Communications Surveys Tutorials,*, vol. 5, no. 2, pp. 2–9, 2003.

[91] B. M. C. Siva Ram Murthy, *Ad Hoc Wireless Networks*. Prentice Hall, 2004.

APPENDICES

Appendix A: Simulation Source Code file 1

```
% Steve Saed
% July 2010
% Simulation of average consensus algorithm in wireless
%sensor network
% File: main.m
clear;
clc;
%*********************************************************
% Simulation Parameters' definition
N = 250;  % Set number of nodes in network
T = 100;  % Set the size of the topology of the
          %simulation (TxT)
R = 10;   % Set the range of communication of each node
E = 0.001;% Set the (Epsilon) difference between max and
          % min state value as a threshold to show that
          % consensus is reached
Th = 0.1:0.01:0.25;%Threshold
Mode = 1;%Communication error mode:
%         0-no error model
%         1-random model
%         2-fading model
%*************************************************************
% Simulation intermediate output
%L - random location coordinates matrix (Nx2)
%A - adjacency matrix based on the L matrix (NxN)
%D - Distance matrix for nodes that are within comm
%range with each other
```

```
%PA -Probabilistic adjacency matrix based on the A matrix (NxN)
%CPA - Current Probabilistic adjacency matrix (NxN)
%weight - metropolis weight matrix
%y - initial state values
%x - intermediate state values
%*****************************************************************
%Make sure a strongly connected graph is generated
connectivity = 0;
while(~connectivity)
    %Generate Random location coordinates for each node in the
    %network
    L = random_location(N,T);
    %Calculate the distances between nodes then construct the
    %adj matrix
    [A,D,Degree] = adj_matrix(N,L,R);
    %Check if graph is strongly connected and find the diameter
    %of the network
    [is_connected,H] =  is_graph_strongly_connected(A);
    connectivity = is_connected;
end
%Average node degree
average_node_degree = mean(diag(Degree));
%Network diameter
network_diameter = max(max(H));
%Initial node state values
y = floor(rand(N,1)*100);
x = y;
%Run consensus Algorithm
%No Error
```

```
[mean_eigen,numIterations,con_x, trace]=
consensus_algorithm(A,D,R,N,Th(1),0,x)
static_output(1) = mean_eigen;
static_output(2) = numIterations;
static_output(3) = mean(y);
static_output(4) = average_node_degree;
static_output(5) = network_diameter;
static_output(6) = 1/(log(1/mean_eigen));%Convergence time
state_value(:,1) = y;
state_value(:,2) = con_x;
save static_trace.dat trace -ascii
%Fading Model
[mean_eigen,numIterations,con_x,trace]=
consensus_algorithm(A,D,R,N,Th(1),2,x)
fading_output(1) = mean_eigen;
fading_output(2) = numIterations;
fading_output(3) = mean(y);
fading_output(4) = average_node_degree;
fading_output(5) = network_diameter;
fading_output(6) = 1/(log(1/mean_eigen));%Convergence time
state_value(:,3) = con_x;
save fading_trace.dat trace -ascii
%Threshold=0.1 simulation
[mean_eigen,numIterations,con_x,trace] =
consensus_algorithm(A,D,R,N,0.1,1,x)
state_value(:,4) = con_x;
threshold_output(1,1) = 0.1;
threshold_output(1,2) = mean_eigen;
threshold_output(1,3) = numIterations;
```

```
threshold_output(1,4) = 1/(log(1/mean_eigen));%Convergence time
save th1_trace.dat trace -ascii
```

Appendix B: Simulation Source Code file 2

```
%*******************************************************
% function [A,D] =  adj_matrix(N,L,R)
% Description : Calculate the distance between each node
%and every other
% node, then generate an adjacency matrix based on the
%communication range of each node
% Arguments   : (1) N - number of nodes in network
%               (2) L - location coordinates matrix
%               (3) R - communication range of each node
% Return Value: Adjacency matrix, distance matrix,
% and degree vector of all the nodes in the network
%*******************************************************/
function [A,D,Degree] =  adj_matrix(N,L,R)
Degree(N,N) = 0;%Initialize the Degree matrix of the network
A = eye(N,N);%initialize adj matrix to identity matrix
% Compute euclidean distance between each pair of nodes
D = squareform(pdist(L));
for i=1:N;
    for j=1:N;
        if((D(i,j) <=R) && i~=j)
            A(i,j) = 1;
            Degree(i,i) = Degree(i,i) + 1;
        end
    end
end
```

Appendix C: Simulation Source Code file 3

```
%************************************************************
% function [mean_eigen,numIterations,average_consensus_value,
average_node_degree,network_diameter] =
%consensus_algorithm(A,D,R,N,Th,Mode,x)
% Description : Consensus iterative algorithm
% Arguments   :
% Return Value:
%************************************************************/
function [mean_eigen,numIterations,con_x,Trace]
=  consensus_algorithm(A,D,R,N,Th,Mode,x)
numIterations = 0;
    while ((max(x)-min(x)) > 1 )
        %No error
        if (Mode == 0)
           weight = metropolis(A);
        %Error Model
        elseif ((Mode==1) || (Mode==2))
            %Determine new adjacency matrix after comm errors are
            %considered
            PA =  prob_adj_matrix(A,D,R,N,Th,Mode);
            % Construct metropolis weight matrix
            weight = metropolis(PA);
        end
        % New Consensus Algorithm Iteration
        newx = weight * x;
        % Check whether a new iteration needs to be run
        %halt_algo = percent_change(E,x,newx);
```

```
        % Upadate node state value
        x = newx;
        % Update the number of iterations of the algorithm
        numIterations = numIterations + 1;
        % Record the state values at each iteration
        % state values (not) converging to consensus
        Trace(:,numIterations) = x;
        %Find all eigen values of eight matrix
        eig_vector = sort(abs(eig(weight)));
        %Record second largest eigen value of the weight matrix
        Eigen_Values(numIterations) =
        eig_vector(length(eig_vector)-1);
    end
    %Node state values after running the average algorithm
    con_x = x;
    %Eigen Values
    mean_eigen = mean(Eigen_Values);
numIterations;
```

Appendix D: Simulation Source Code file 4

```
%*********************************************************
% function [C,H] =  is_graph_connected(A)
% Description : Check if the adjacency matrix represents
%a connected graphFor a graph to be strongly connected,
%there must exist a path between any two given nodes.
%The graph is assumed to be undirected because of two
%nodes are within comm range from each other, then they
%theoretically can communicate with each other
% Arguments   : (1) A - Adjacency matrix of the network
% Return Value: (1) is_connected - Bool value to indicate
                    if the graph is connected
%               (2) H - Hop-matrix, a matrix with the same
%                   size as A, eachentry (i,j) gives the
%                   number of hops needed to go from  i to j
%*********************************************************/
function [is_connected,H] =  is_graph_strongly_connected(A)
%Check size of the matrix
[n1 n2] = size(A);
if (n1~=n2)
    return;
end
%Initialization
is_connected = 1;
%One hop to each neighboring node
H(n1,n2) = 0;
%Check if there is a path from every node to every other node
for i=1:n1
```

```
    for j=1:n2
        %find shortest path between any two given nodes
        H(i,j) =
        graphshortestpath(sparse(A),i,j,'Method','BFS');
        if( H(i,j)==Inf)
            is_connected = 0;
            break;
        end
    end
end
```

Appendix E: Simulation Source Code file 5

```
%************************************************
% function [L] = random_location(N,T)
% Description : Generate random coordinates for the
% N network agent within the square topology TxT
% Arguments   : (1) N - number of nodes in network
%               (2) T - TxT size of square topology
% Return Value: Nx2 vector of random location
% coordinates of each node
%************************************************/
function [L] = random_location(N,T)
L = rand(N,2)*T;
```

Appendix F: Simulation Source Code file 6

```
%************************************************
% function [PA] =  prob_adj_matrix(A,D)
% Description : Calculate the probability of
% communication between nodes within comm range
% of each other based on the distance between them.
% Arguments   : (1) D - distance matrix between nodes
%               (2) R - communication range
%               (3) N - number of nodes in the network
%               (4)Mode - 0-fading model; 1- random model
% Return Value: Probabilistic Adjacency matrix of the network
%*******************************************************/
function [PA] =  prob_adj_matrix(A,D,R,N,Th,Mode)
PA = eye(N,N);
for i=1:N;
    for j = 1:N;
        if ((A(i,j)==1) && (i~=j))
            %Generate random number for Bernouilli experiment
            rand_ber = rand(N,N);
            %Random model/Approximated fading model
            if(Mode == 1)
                comm_prob = Th;
            %Fading Model
            elseif (Mode == 2)
                comm_prob = exp(-5*D(i,j)/R)-(D(i,j)/R)*exp(-5);
            end
            %Bernouilli experiment to determine if link is connected
            if (rand_ber(i,j) >= comm_prob)
```

```
                PA(i,j) = 1;
            elseif (rand_ber(i,j) < comm_prob)
                PA(i,j) = 0;
            end
        end
    end
end
```

Appendix G: Simulation Source Code file 7

```
%*****************************************************
% function [halt_algo] =  percent_change(E,x,newx)
% Description : determine whether the algorithm should
%               be stopped based on percent change between
%               current and previous state values
% Arguments   : (1) E - epsilon
%               (2) x - previous state value
%               (3) newx - current state value
% Return Value: halt_algo - bool value to determine if
%               the algorithm should be stopped
%*****************************************************/
function [halt_algo] =  percent_change(E,x,newx)
x1 = x;
x2 = newx;
x3 = x2 - x1;
x3 = x3./x1;
%Set default value to 1 - halt algorithm
halt_algo = 1;
for i=1:length(x3)
    if (abs(x3(i)) > E)
        %If one state value changed by more than E,
        %continue running the algorithm
        halt_algo = 0;
    end
end
```

Appendix H: Simulation Source Code file 8

```
%*****************************************************
% function [W] = metropolis(A)
% Description : Construct metropolis weight matrix
% Arguments   :
%(1) CPA - Current Probabilistix Adjacency matrix
% Return Value: Mertropolis weight matrix
%*****************************************************/
function [W] = metropolis(CPA)
[n,m] = size(CPA);
%Make-sure CPA matrix is square
if n ~= m
    return
end
%Initialization
W = zeros(n,n);
in_degree = zeros(n);
out_degree = zeros(n);
%construct in-degree, out-degree vectors
for i=1:n;
   %find in-degree of each node in the network
   in_degree(i) = length(find(CPA(i,:)))+1;
   %find out-degree of each node in the network
   out_degree(i) = length(find(CPA(:,i)))+1;
end
%Calculate metro-weights except self-weights
for i=1:n;
    for j=1:n;
```

```
        if ((i~=j) && (CPA(i,j) ~= 0));
            W(i,j) = 1/(max(in_degree(i),out_degree(j)));
        end
    end
end
%Calculate metropolis self-weights
for i=1:n;
    W(i,i) = 1-sum(W(i,:));
end
```

Appendix I: Simulation Source Code file 9

```
%************************************************
% dskim- consensus
% function [answer] = is_doubly_stochastic(A)
% Description : Construct metropolis weight matrix
% Arguments   : (1) A - metropolis weight matrix
% Return Value: 1 if weight matrix is doubly stochastic,
%otherwise 0
%************************************************/
function [answer] = is_doubly_stochastic(A)
[n,m]=size(A);
if n ~= m
    answer = false;
    return
end
h = sum(A);
v = sum(A');
u = ones(1,n);
if h ~= u
    answer = false;
    return
end
if v ~= u
    answer = false;
    return
end

answer = true;
```