AN ADAPTIVE, ENERGY-AWARE AND DISTRIBUTED FAULT-TOLERANT TOPOLOGY-CONTROL ALGORITHM FOR HETEROGENEOUS WIRELESS SENSOR NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FATIH DENIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

AUGUST 2016

Approval of the thesis:

AN ADAPTIVE, ENERGY-AWARE AND DISTRIBUTED FAULT-TOLERANT TOPOLOGY-CONTROL ALGORITHM FOR HETEROGENEOUS WIRELESS SENSOR NETWORKS

submitted by **FATIH DENIZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering**

Prof. Dr. Adnan Yazıcı
Supervisor, **Computer Engineering Dept., METU**

Dr. Hakkı Bağcı
Co-supervisor, **Mobile Action**

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering Dept., METU

Prof. Dr. Adnan Yazıcı
Computer Engineering Dept., METU

Assoc. Prof. Dr. İbrahim Körpeoğlu
Computer Engineering Dept., Bilkent University

Asst. Prof. Dr. Mustafa Sert
Computer Engineering Dept., Başkent University

Assoc. Prof. Dr. Ahmet Oğuz Akyüz
Computer Engineering Dept., METU

Date:

ABSTRACT

AN ADAPTIVE, ENERGY-AWARE AND DISTRIBUTED FAULT-TOLERANT TOPOLOGY-CONTROL ALGORITHM FOR HETEROGENEOUS WIRELESS SENSOR NETWORKS

Deniz, Fatih
Ph.D., Department of Computer Engineering
Supervisor : Prof. Dr. Adnan Yazıcı
Co-Supervisor : Dr. Hakkı Bağcı

August 2016, 96 pages

Wireless sensor networks(WSNs) are being used in numerous fields, such as battlefield surveillance, environmental monitoring and traffic control. They are typically composed of large numbers of tiny sensor nodes with limited resources. Because of their limitations and because of the environments they are being used, there are problems unique to WSNs. Due to the error-prone nature of wireless communication, especially in harsh environments, fault-tolerance emerges as an important property in WSNs. Also, because of the battery limitations, solutions to reduce energy consumption and prolong network lifetime are quite valuable. In this thesis, we propose two algorithms, namely Adaptive Disjoint Path Vector (ADPV) and Minimum Supernode Disjoint Path Vector (MSDPV), for heterogeneous WSNs. In this heterogeneous model, we have resource-rich supernodes as well as ordinary sensor nodes that are supposed to be connected to the supernodes. MSDPV algorithm considers the desired fault-tolerance degree and the positions of ordinary sensor nodes to determine optimal number of supernodes and their locations. It provides a novel optimization based on the well-known set-cover problem. ADPV is an adaptive, energy-aware and distributed fault-tolerant topology-control algorithm. Unlike the static alternative Disjoint Path Vector (DPV) algorithm, the focus of ADPV is to secure supernode connectivity in the presence of node failures, and ADPV achieves this goal by dynamically adjusting the sensor nodes' transmission powers. The ADPV algorithm involves two phases: a single initialization phase, which occurs at the beginning, and restoration phases, which are invoked each time the network's supernode connectivity is broken. Restoration phases utilize alternative routes that are computed at the initialization phase by the help of a novel optimization based on the well-known set-packing problem. Through extensive simulations, we demonstrate that ADPV is superior in preserving supernode connectivity. In particular, ADPV achieves this goal up to a failure of 95% of the sensor nodes; while the performance of DPV is limited to 5%. In turn, by our adaptive algorithm, we obtain a two-fold increase in supernode-connected lifetimes compared to DPV algorithm.

Keywords: Topology control, fault-tolerance, energy efficiency, prolonged network lifetime, *k*-connectivity, heterogeneous wireless sensor networks, maximum set packing

ÖZ

HETEROJEN KABLOSUZ SENSÖR AĞLARI İÇİN UYARLANABİLİR, ENERJİ SEVİYESİ FARKINDA VE DAĞITIK HATA TOLERANSLI TOPOLOJİ KONTROL ALGORİTMASI

Deniz, Fatih
Doktora, Bilgisayar Mühendisliği Bölümü Bölümü
Tez Yöneticisi : Prof. Dr. Adnan Yazıcı
Ortak Tez Yöneticisi : Dr. Hakkı Bağcı

Ağustos 2016 , 96 sayfa

Kablosuz sensör ağları savaş gözetimi, çevresel izleme ve trafik kontrolü gibi birçok alanda kullanılmaktadır. Bu ağlar, algılama, işleme ve aktarma yeteneğine sahip çok sayıda küçük sensör düğümlerinden oluşmaktadır. Bu ağları oluşturan sensör düğümleri genellikle herhangi bir altyapısı bulunmayan bir ortamda kendi kendilerine organize bir şekilde çalışmakta ve kendilerine verilen görevi işbirliğiyle yerine getirmektedir. Bu ağların en büyük üç problemi sensör düğümlerinin sınırlı kaynakları sebebiyle enerjilerinin tükenmesi, çalışma ortamlarının zorlu koşulları nedeniyle bazı sensor düğümlerinin bozulması ve veri iletişiminde yaşanan problemler olarak sayılabilir. Bu nedenle, bu problemlere çözüm olarak önerilen ağ ömrünü uzatan algoritmalar ve hata toleranslı topoloji kontrol algoritmaları yüksek öneme sahiptir. Bu tezde heterojen kablosuz sensör ağları için Adaptif Ayrık Yol (ADPV) ve Minimum Süper Düğüm Ayrık Yol (MSDPV) isminde iki algoritma sunulmaktadır. Temel alınan heterojen mimaride kaynak bakımından zengin süper düğümler ve süper düğümlere bağlı olması gereken sınırlı kaynağa sahip sıradan sensör düğümleri bulunmaktadır. MSDPV algoritması hedeflenen hata tolerans seviyesi ve sıradan sensör düğümlerinin konumlarına göre en uygun süper düğüm sayısı ve konumlarını hesaplamaktadır. Bu hesaplama sırasında bilinen matematiksel bir optimizasyon problemi olan maksimum küme kapsama algoritması kullanılmaktadır. ADPV algoritması ise uyarlanabilir, enerji seviyesi farkında ve dağıtık bir hata toleranslı topoloji kontrol algoritmasıdır. Sensör düğümlerinin herhangi bir sebeple ölmesi ve ağın hata toleransının belirli bir seviyenin altına inmesi durumunda Adaptif Ayrık Yol (ADPV) algoritması sensör düğümlerinin iletim güçlerini ayarlamakta, ağdaki hata toleransının belirli bir seviyenin altına inmemesini garanti etmekte ve enerji seviyesi farkındalığı sayesinde de yükü dengeli bir şekilde dağıtabilmektedir. ADPV algoritması iki evreden oluşmaktadır: bilgi toplama ve alternatif yollar belirleme adımlarından oluşan başlangıç aşaması ve sensör düğümlerinin süper sensörler ile olan bağlantıları koptuğu durumlarda çalışan restorasyon aşamaları. Restorasyon aşamaları sırasında kullanılan alternatif yollar, bilinen matematiksel bir optimizasyon problemi olan maksimum küme paketleme algoritması kullanılarak oluşturulmaktadır. Gerçekleştirilen simülasyonlar, ADPV'nin ağın bağlantılı bir şekilde çalışmasındaki başarısını göstermektedir. DPV algoritması ile ağdaki düğümlerin en fazla %5'i öldüğünde ağ bağlantısı koparken, ADPV algoritması ile sensör düğümlerinin %95'i ölene kadar ağ bağlantılı bir şekilde çalışabilmektedir. Bunula birlikte, ADPV algoritması, DPV algoritmasına göre iki kat daha fazla bağlantılı bir ağ ömrü elde edebilmektedir.

Anahtar Kelimeler: Heterojen Kablosuz Duyarga Ağları, Topoloji Kontrol, Hata Toleransı, Uzun Süreli Ağ Ömrü, Ayrık Yollar

To my family..

ACKNOWLEDGMENTS

First and foremost I want to thank my supervisor Prof. Dr. Adnan Yazıcı for all the support and encouragement he gave me. His guidance and positive approach throughout this time period made possible to complete this work.

I would like to express my gratitude and profound respect to my co-advisor Dr. Hakkı Bağcı. His invaluable ideas, suggestions and help have been essential to my research. I appreciate all the time he has spent for the development of my thesis.

I acknowledge with thanks and appreciation the members of thesis monitoring comittee, Assoc. Prof. Dr. İbrahim Körpeoğlu and Assoc. Prof. Dr. Ahmet Oğuz Akyüz. Their advice helped me to move forward in this study.

I thank to the members of the thesis jury, Prof. Dr. Ahmet Coşar and Asst. Prof. Dr. Mustafa Sert for reviewing and evaluating my thesis.

I would also like to thank to TÜBİTAK (The Scientific and Technological Research Council of Turkey) for their support of this graduate study with the BIDEB 2211 support program.

Special thanks to my friend Dr. Enver Kayaaslan for his support, help and invaluable contributions to complete this work.

I would also like to express my thanks to my friends, especially Dr. Hakan Bağcı and Zafer Selçuk Aygın, for their invaluable friendship, support and joy they bring to my life.

Finally, I would like to express my thanks to my parents, sisters, my wife and my son, making me who I am now with their love, trust, understanding, encouragement and every kind of supports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Wireless sensor networks (WSNs) are typically composed of large numbers of tiny sensor nodes that are capable of sensing, processing and transmitting data over wireless channels. Such networks can be used in numerous fields, such as battlefield surveillance [26, 22, 82], environmental monitoring [105, 59, 32] and traffic control [65, 92, 14]. Sensor nodes collaborate in a distributed, autonomous and self-organized manner to accomplish a certain task, usually in an environment with no infrastructure.

Sensor nodes in WSNs should be low-cost and should have small form-factor. This restricts sensor nodes in many ways as they have limited energy, short transmission range, relatively slow CPU and small memory. These limitations bring out many challenges unique to WSNs, such as very low power consumption. Since sensor nodes are battery powered and these batteries are usually not rechargeable, coming up with solutions that reduce energy consumption and prolong network lifetime are very important. Numerous studies address this problem [87, 28, 4, 12] in literature. According to Li and Mohapatra [50], 90% of a sensor network's energy is still available after first node dies. Despite this substantial amount of remaining energy, the existence of highly-loaded and bottleneck nodes cause early network demise. There are numerous studies that address balancing energy consumption among nodes to ensure that all nodes will run out of energy at about the same time [103]. Low-energy Adaptive Clustering Hierarchy (LEACH) [34] is a well-known early study that uses dynamic transmission ranges to better balance the load and prolong network lifetime. There are also some recent studies which reestablish lost connectivity by adjusting transmission ranges. CoRAD [38] and RESP [88] can be listed as some of those studies. With the recent developments in the hardware of WSNs, dynamic transmission range assignment has become even more effective [69].

Another approach to prolong network lifetime is to form a heterogeneous WSN architecture with supernodes which act as alternative gateways to the monitoring center of a WSN [7]. That means it is enough if the sensory data can reach any one of these supernodes. A heterogeneous WSN with supernodes is known to be more reliable and has longer network lifetime than the homogeneous counterparts without supernodes. Yarvis et al. [97] reported that heterogeneity can triple the average delivery rate and provide a five-fold increase in the network lifetime if supernodes are deployed carefully.

Fault-tolerance is another critical issue in WSNs. Due to the error-prone nature of wireless communication, links may fail, packets can get corrupted or congestion may occur [93, 62]. There are also factors that cause long-term faults in sensor nodes, such as energy depletion, hardware failure, link breaks, malicious attacks. Multi-hop communication multiplies the chances of faulty incidents for a packet stream traveling from a source to a destination. Therefore, fault-tolerance methods, including fault-tolerant topology control, are essential for improving WSN reliability as well as network lifetime.

As stated by Liu et al. [55], most existing works on fault-tolerant topology-control aim to obtain k-vertex connectivity between any two sensor nodes, where the topology is guaranteed to remain connected until the failure of the kth sensor node.

In this study, the focus is on two-tiered heterogeneous WSNs, where the network consists of two different types of nodes: resource-rich supernodes and simple sensor nodes with limited battery power. In this network model, sensor nodes are connected to the set of supernodes via multi-hop paths. To reflect this asymmetry, [10] proposes k-vertex supernode connectivity, where each sensor is connected to at least one supernode by k vertex-disjoint paths. In such topologies, the sensor nodes remain connected to the supernodes as long as at most $k - 1$ sensor nodes fail.

Most studies on fault-tolerance propose static solutions, that is, they do not adapt the topology to the changing network conditions. Bagci et al. [6] propose a static algorithm called the Disjoint Path Vector (DPV) to optimize total transmission power for a given *k*-vertex supernode-connected network. That study does not consider residual battery energy and disregards the unbalanced load distribution on sensor nodes. As a result, *k*-vertex supernode connectivity is achieved but may not be preserved for a sufficient amount of time.

Another critical issue in heterogeneous WSNs with supernodes is the number of supernodes and their possible locations. In the literature [10, 6, 18] addresses fault-tolerance, connectivity and heterogeneity at the same time. However, they mainly concentrate on the transmission powers of the sensor nodes and share a common drawback, that is, they all assume that a certain number of supernodes are initially deployed without considering where sensor nodes are located.

In this thesis, we propose two algorithms, namely Adaptive Disjoint Path Vector (ADPV) and Minimum Supernode Disjoint Path Vector (MSDPV), for heterogeneous WSNs.

ADPV is a novel adaptive and distributed topology-control algorithm, which efficiently constructs a *k*-vertex supernode-connected network topology and adapts the topology to node failures, which in turn increases network lifetime. The contribution is two-fold. First, the residual battery power levels of individual sensor nodes are considered to prolong *k*-vertex supernode connectivity. Second, an adaptive solution is proposed to restore, if necessary, *k*-vertex supernode connectivity after a node failure.

MSDPV, on the other hand, has a completely different aim. It aims to locate minimum number of supernodes into a network of sensor nodes, so that the resulting topology becomes *k*-vertex supernode-connected. MSDPV is also a novel energy-aware and distributed algorithm and it considers the desired fault-tolerance degree and the positions of ordinary sensor nodes to determine optimal number of supernodes and their locations. Through extensive simulations we demonstrate that MSDPV achieves up to 40% improvement in supernode-connected lifetimes compared to random and uniform distribution of supernodes. When MSDPV algorithm is used together with ADPV algorithm, both supernode locations and transmission powers of the sensor nodes is optimized and a more efficient solution for heterogeneous WSNs is being formed.

The remainder of this thesis is organized as follows: Chapter 2 gives a brief overview of WSNs and presents a background for our research. Chapter 3 discusses the related studies in the literature that address fault-tolerance and connectivity restoration in heterogeneous WSNs. In Chapter 4, we present our proposed adaptive topology-control solution and in Chapter 5, we present our solution to supernode placement problem that calculates minimum number of resource-rich supernodes and their possible locations. The results for simulation experiments are presented in Chapter 6. Finally, Chapter 7 concludes the thesis.

CHAPTER 2

BACKGROUND

2.1 Overview of Wireless Sensor Networks

Wireless sensor networks (WSNs) consist of tiny sensor nodes deployed over a geographical area for monitoring physical phenomena like temperature, humidity, vibrations, and so on [3]. In general, sensor networks support new opportunities for the interaction between humans and their physical world and are used in a wide range of areas, such as military applications, public safety, medical applications, environmental monitoring, commercial applications. Due to the recent technological advances in low-power digital and analog circuitry, wireless communications and microelectronic systems, the manufacturing of small, low-cost and multifunctional sensors has become technically and economically feasible. Typically, a sensor node is a tiny device that includes four basic components: a sensing unit for acquiring data from the environment, a processing unit to process the gathered data, an antenna used for wireless communication and data transmission, and a battery unit to supply the required energy.

2.2 Heterogeneous WSNs

Usually, there are two types of sensor nodes that build up the architecture of homogeneous WSNs. These are sensor nodes and sink nodes.

- Sensor nodes are the main entities that form the WSN architecture. Their main objectives are making discrete, local measurement about the environment surrounding them, forming a wireless network by communicating over a wireless medium, collecting data from its neighbors and routing it to the sink nodes.

- Sink nodes are also known as base stations that have higher resources in terms of battery power, computation capability, communication range and storage capacity. This also increases their costs and only a few sink nodes exist in the whole WSN. Collected data from the sensor network is routed back to the sink nodes and sinks usually inform user application regarding the gathered data.

Contrary to homogeneous WSNs, heterogeneous WSNs consist of sensor nodes with different capabilities, such as different computing power, sensing range, communication range or energy capacity [21]. These differences make topology control and deployment of sensor nodes in heterogeneous WSNs more complex than homogeneous counterparts, but provides more flexibility in the deployment process [94]. Also, a heterogeneous WSN is known to be more reliable and has longer network lifetime than the homogeneous counterparts. Yarvis et al. [97] reported that heterogeneity can triple the average delivery rate and provide a five-fold increase in the network lifetime.

Nodes with superior capabilities compared to ordinary nodes are called supernodes. Supernodes are also called as actors in wireless sensor and actor networks (WSANs) and they are usually well-equipped nodes in terms of battery power, computation capability and communication range [68]. They are usually located inside the sensor field and they act as alternative gateways to the monitoring center of WSNs [7]. That means it is enough if the sensory data can reach any one of the supernodes. Heterogeneous deployments usually contain a mixture of resource-rich supernodes and much cheaper sensor nodes with limited computation and communication capabilities. In this way, a mixed deployment achieves a balance between performance and cost [94].

There are also some other types of heterogeneity such as nodes with different transmission powers, nodes with different battery supplies, relay nodes together with the regular sensor nodes. Relay nodes are quite expensive sensor nodes and generally used for forwarding the data traffic originated from the sensor nodes to the sink nodes and most of the studies regarding heterogeneity address this type of sensor networks.

In this study, the focus is on two-tiered heterogeneous WSNs, where the network consists of two different types of nodes: resource-rich supernodes and simple sensor nodes with limited battery power. In this network model, sensor nodes are connected to the set of supernodes via multi-hop paths. To reflect this asymmetry, [10] proposes k-vertex supernode connectivity, where each sensor is connected to at least one supernode by k vertex-disjoint paths. In such topologies, the sensor nodes remain connected to the supernodes as long as at most $k - 1$ sensor nodes fail.

In this study, we apply two types of heterogeneity. First, we use supernodes together with sensor nodes to form a hierarchical network structure. Second, we adapt the transmission powers of the sensor nodes to the changing network topology and form a heterogeneous network structure with different transmission ranges. In this way, we aim to prolong connected lifetime of the network, and at the same time preserve fault-tolerance property of the network.

2.3 Design Factors and Requirements of WSNs

A sensor node by itself has severe resource constraints, such as limited battery power, computation and communication capabilities. However, a group of sensors collaborating with each other can accomplish much bigger tasks efficiently.

Sensor networks may contain hundreds to thousands of sensing nodes that are aiming to send its sensed data to a base station and it is totally desirable to make these nodes as energy efficient as possible. Also, network protocols must be designed to achieve fault-tolerance in presence of unexpected node failures. Moreover, bandwidth of wireless channels is very limited and shared by all sensors in the network. Thus, routing protocols and topology control algorithms should be able to perform local collaboration to reduce bandwidth requirements.

Also, sensor nodes have limited processing capability, storage capacity and communication range. In order to make an application for WSNs, it is necessary to consider all these limitations which make the work very challenging. In the literature, many design factors have been proposed by many researchers. In this section, we intend to describe and give more detail on some of these limitations, design factors and requirements of algorithms and protocols for WSNs. In this study, we mainly focus on connectivity, fault-tolerance and power consumption design factors, but also consider other limitations like hardware constraints, scalability, dynamic network structure and self-configuration.

2.3.1 Power Consumption

In mobile and ad hoc networks, power consumption is not the most important consideration since the power resources can be recharged or replaced by the user. For WSNs, sensor nodes can only be equipped with a limited power source. In most applications, even the replacement of power resources is impossible. Usually sensor nodes are deployed in a hostile environments, which makes it impossible or at least inconvenient to recharge the battery. On the other hand, lifetime requirements of WSN applications can be weeks to months or even years [71]. Therefore, energy should be used very sparingly and solutions to prolong network lifetime in WSNs is of considerable importance.

In the literature, there are several approaches to minimize the network's total energy consumption and improve network lifetime. Some of these approaches are adjusting transmission powers [9], developing energy-efficient MAC or routing protocols [77, 57, 64], and putting some sensor nodes into sleep mode and using only a necessary set for sensing and communication [86].

For most applications, algorithms proposed to minimize power consumption also needs to consider quality-of-service up to some point. Since dis-functioning of few nodes can also cause significant topological changes and might require re-routing of packets which will consume more power, while making a trade-off, more importance and higher privilege is given to power consumption, but still with the aim of reducing overall power consumption, connectivity and fault-tolerance requirements are also needed to be considered.

2.3.2 Connectivity

In WSNs, the information collected by the sensor nodes is needed to be sent to data collection centres and this is only possible if there exists a path from each node to the collection centre. Therefore, connectivity of a WSN is also one of the most important design factors and is usually studied by considering a graph associated with that WSN. A network is called connected if its associated graph is connected and a graph is connected if and only if there exists a path between any pair of its vertices [20].

In WSNs, if there exists a path (single or multi-hop) between a sensor node to any of the sink nodes, then the sensor node is said to be connected. If every (alive) sensor node in the network is connected, then the network is also considered as connected. In WSNs, each sensor node has a communication range that determines the maximum distance it can transmit data and sensor nodes communicate with the nodes within their range. In WSNs, with the increase in transmission power, communication range also increases and this leads to the increased probability of network connectivity. However, the increase in power consumption is not directly proportional, but much higher than the increase in the communication range. Also, larger power results in severe interference within the network. On the other hand, reducing the communication range reduces the probability of connectivity and increases the number of hops required to reach the destination. Therefore, while proposing topology control algorithms for WSNs this tradeoff should be carefully studied.

2.3.3 Coverage

WSNs are used to monitor a given field of interest for changes in the environment and coverage is usually defined as a measure of how well the sensor network monitors the field of interest [60]. The sensor node's view of the environment that it is situated in is limited both in range and in accuracy. This means the ability of sensor nodes to cover physical area of the environment is limited.

The coverage problem for WSNs has been studied extensively in recent years, especially when combined with connectivity and energy efficiency [24]. According to different objectives and application requirements there are two types of coverage problems: area and border coverage [96]. Area coverage problem asks for the minimum number of sensor nodes when the region is covered by a connected WSN. Similarly, border coverage requires every point on the border line to be covered by a connected WSN.

2.3.4 Fault-tolerance

Another very critical design factor in WSNs is fault-tolerance. Causing from the lack of power, environmental interference or physical damage, some sensor nodes may be blocked or failed to work. Fault-tolerance is defined as the system to work correctly under such circumstances and sustain sensor network functionalities without any interruption [46].

Fault-tolerance can be considered for both coverage and connectivity problems. For connectivity, in order to establish fault-tolerance it is useful to consider stronger forms of connectivity, such as k-connectivity, in which the network remains connected even if $k - 1$ nodes are removed. If a network is k-connected ($k \geq 2$), it has better fault-tolerance than if it is merely 1-connected. It is stated that ensuring k-connectivity extends the network lifetime if nodes fail at random times [49]. Similarly for coverage, it is desirable for each point to be tracked by at least k sensing nodes for establishing stronger coverage.

Fault-tolerance requirement is subject to change according to the environmental conditions of the sensor nodes. For example, if the sensor nodes are deployed in a house environment to keep track of sound level, it is not necessary for the system to have a high-level of fault-tolerance, since this kind of sensor networks is not easily damaged or interfered. Therefore, during the design phase of WSNs, fault-tolerance degree of the network should be determined according to application needs and environmental conditions.

2.3.5 Scalability

Depending on the application, number of sensor nodes deployed in the sensing area may be hundreds or even thousands. The routing scheme should not be dependent on the number of sensor nodes and should be able to work with huge number of sensor nodes. Also, sensor network routing protocols should be scalable enough to respond to density of events in the environment. For example, when the human body is being tracked, the density of the sensor nodes and generated event packets will be extremely high. Therefore, density of the nodes should also be considered in designing the scalability of the sensor networks.

2.3.6 Hardware Constraints

Sensor node consists of four main components: a sensing units, processing unit, transmission unit, and power unit. They may also have application-dependent additional components such as position/location finding systems, power generator, and mobilizer. Sensing units are usually composed of two subunits: Sensors and ADC (Analog to Digital Converter). Analog signals produced by sensors based on the observed phenomenon are converted by ADC to digital signal and fed into the processing unit to be processed. Processing unit, generally associated with storage unit, manages the procedures that make the sensor node collaborate with other nodes to perform the assigned sensing tasks. Transmission unit connects the sensor node to the network. Power unit may be supported by a power scavenging such as solar cells. Sometimes, a mobilizer is needed to move sensor nodes to carry out the assigned tasks. Therefore, while designing an algorithm or protocol in WSNs, all these constraints should be taken into account.

2.3.7 Data Aggregation/Data Fusion

In WSNs, as sensor networks made of large number of sensor nodes; network can easily be congested with the flooding information. A solution to data congestion in sensor networks is to use computation to aggregate or fuse data within the sensor nodes, and then transmit only the aggregated data to the controller. Especially for dense deployments like body area networks, where a huge upstream traffic can be generated, data aggregation is a quite important design factor [48].

2.3.8 Security

WSNs suffer from many constraints, including limited energy resources, computation capability, and small memory. These constraints make security in WSNs quite challenging. For example, the limited energy and processing power makes use of public key cryptography in WSNs almost impossible [90]. Some of the security threats in WSNs are categorised as follows [75]: passive information gathering, false node, node outage, supervision of a node, message corruption, and denial of service. Especially for mission critical applications, like military applications, considering security in WSNs is quite essential [39].

2.3.9 Self-Configuration

Because of the unattended work of sensor nodes in dynamic environments; self-configuration to establish a topology that supports communications under severe energy constraints is quite important. Also, since the densely deployed sensor nodes in a sensor field may fail due to many reasons (e.g., lack of energy, physical destruction, environment interference, communications problem, inactivity, etc.) and new nodes may join the network, sensor nodes should self-organize to adopt to the changing environment and topology. Thus, self-configuration is also one of the critical design issues in almost all WSN applications.

2.3.10 Network Dynamics

In many applications, the movement of sensor nodes or the base station (sink) is quite important. This means sensor nodes are mobile nodes (i.e., not stationary as assumed by many of network architectures). There are several recent studies regarding mobile sink nodes [43, 44]. In this approach, sink nodes move within the sensing field, collect data from the sensor nodes and bring them back to the base. In this way, due to reduced multi-hop communication, significant energy-saving and prolonging network lifetime is possible.

Also, in addition to using mobile nodes, the specific sensed phenomenon may either be dynamic (e.g., target detection/tracking applications) or stationary (e.g., forest monitoring) depending on the applications. Therefore, while designing an algorithm for WSNs, considering network dynamics is also quiet important.

2.3.11 Quality of Service

For some applications, data delivery within a bounded latency have great importance. If the application does not comply with the time constraints, the sensed data that delivered after certain latency will be useless. In other applications the conservation of power is more important than the quality and latency of the delivered data. Hence; there is a tradeoff between the quality of service/the quality of data sent and the energy conservations or consumption depending on the applications. For instance, although mobile sinks that collect sensed data can significantly prolong network lifetime, because of the low movement speed of mobile nodes, their usage is limited for time-constraint applications [98]. During our studies, we have given more importance to the energy conservation, but we also ensure some level of quality of service with the constraints like maximum-hop-count for the longest possible paths.

2.4 Topology Control in WSNs

Unit Disk Graph (UDG) is a commonly used model to reflect the behaviour of sensor nodes. In this model, it is assumed that all the sensor nodes has a fixed radius for transmission ranges and all the nodes within their range are neighbours [47]. Although, UDG is a simple yet effective approach, it consumes too much energy, which is the scarcest resource in WSNs. Topology control is a method used to close this gap of UDG using the multiple energy level support of the sensor nodes. The current sensor nodes commonly support multiple energy levels and under different energy levels, transmission ranges of the sensor nodes also vary [51]. Topology control is defined as controlling the set of neighbour nodes to forward the messages to by adjusting transmission ranges (by adjusting energy levels) and/or by selecting specific nodes [89].

The primary goal of topology control in WSNs is to prolong network lifetime while preserving connectivity. Besides reducing the energy consumption, topology control also reduces radio interference, which increases network's traffic carrying capacity. In the literature there are several topology control algorithms [3, 51, 91, 52]. According to their aims topology control algorithms are categorized into two groups [74]:

- In the *homogeneous* topology control, all the sensor nodes have the same transmission range and relevant literature asks for the minimum common value for the transmission range that keeps the network connected.
- In the *nonhomogeneous* topology control, sensor nodes are allowed to have different transmission ranges, but the resulting network still needs to be connected. In this study, we propose a nonhomogeneous topology control algorithm that considers residual battery power levels of the sensor nodes to determine their transmission ranges.

2.5 Adaptive Approaches in WSNs

There are several approaches in WSNs that adaptively change network topology during networks' lifetime [54, 29].

Low-energy Adaptive Clustering Hierarchy (LEACH) [34] is a well-known early study that uses adaptive transmission ranges to better balance the load and prolong network lifetime. LEACH is a clustering based protocol that utilizes randomized rotation of local cluster heads that are used as routers to the sink. According to the changing cluster heads, sensor nodes adjust their transmission power to reach its new cluster head. The main purpose of the protocol is evenly distributing the energy load among all sensor nodes. Since sensor nodes transmit their data only to their cluster head, rather than all sensor nodes, LEACH enables significant energy savings. LEACH enables scalability and robustness by using localized coordination and it estimates optimal number of cluster heads to be 5% of the total number of sensor nodes.

In order to reduce amount of information carried between sensor nodes, LEACH also uses data fusion in the cluster heads.

One other example to adaptive approaches is Adaptive Self-Configuring sEnsor Networks Topologies (ASCENT) [11]. ASCENT is an adaptive solution for making sure network is connected throughout its lifetime. It is build on the notion that, as density increases, only a subset of the nodes is necessary to establish a routing forwarding backbone and therefore, initially only some nodes are active. Passive nodes listen to packets but do not transmit. If the number of active nodes is not large enough, the sink node may experience a high message loss from sources. The sink then starts sending help messages to solicit neighbouring nodes that are in the passive state to join the network by changing their state from passive to active. This process continues until message loss rate experienced by the sink is below a predefined threshold. The process will restart when some future network event like node failure or environmental effect causes packet loss again.

2.6 Placement of Sink Nodes in WSNs

In WSNs, sensor nodes can be placed carefully to engineered positions or thrown in bulk to random positions [3]. Each of these methods has their own advantages and disadvantages. Placement of the sensor nodes obviously has an important effect on the resource management of WSNs [19]. On the other hand, it is not always feasible to place thousands of tiny sensor nodes to known locations. First, usually number of sensor nodes to be deployed is very large and second, the application environment is usually not completely accessible. Because of these reasons, especially for the ordinary sensor nodes, it is usually preferable to sacrifice some more resources than work on the placement problem.

Most of the existing studies on placement of sensor nodes aim to minimize number of required sensor nodes, and at the same time preserve application specific constraints like coverage, connectivity or fault-tolerance [30, 41, 79]. In this study, we consider determining the minimum number of required supernodes and placing them to known locations according to randomly distributed sensor node locations. We also have the constraints of preserving connectivity and fault-tolerance factors.

Our approach is different than the relay node placement approaches that aim to restore connectivity. The aim in the relay node placement algorithms are connecting sensor nodes within each other or sink nodes using relays. In this study we aim to place supernodes, where the traffic is destined to, so that network becomes k-vertex supernode-connected.

In the literature there are several studies that address multiple sink placement problem [16, 78, 72].

CHAPTER 3

RELATED WORK

In this chapter we give a brief overview of some of the prominent recent work addressing fault-tolerance and connectivity restoration in WSNs. We also give a brief overview of the DPV algorithm [6].

Fault-tolerance techniques can be categorized into four [83]:

- *Prevention* attains network connectivity and establishes redundant links/nodes when necessary.
- *Detection* monitors traffic and sends alerts when any indication of fault happens, such as a decrease in packet delivery rate, which would imply a packet loss, interruption, or delay.
- *Isolation* diagnoses and identifies the alert.
- As for *recovery*, after detecting and identifying the fault, the system should be able to recover in either a centralized or distributed manner. Note that due to the nature of WSNs it is essential for the recovery scheme to be a distributed method.

The replication and redundancy of components prone to failure is the most commonly used method for fault prevention and recovery [55]. For instance, if some nodes have problems and fail to sense the environment, the redundant nodes in the vicinity can still provide data. Keeping redundant links or multiple paths also provides fault-tolerance when some communication links are broken due to node failures or communication errors.

3.1 Connectivity Restoration in WSNs

Due to the error-prone nature of wireless communication, links may fail, packets can get corrupted or congestion may occur [93, 62]. There are also factors that cause long-term faults in sensor nodes, such as energy depletion, hardware failure, link breaks, malicious attacks. As a result of these failures, nodes may leave some areas uncovered and reduce the accuracy of the collected data. However, the most serious consequence is when the network gets disconnected. Since losing connectivity prevents data exchange and coordination among sensor nodes, it has a very negative effect on the applications. Therefore, restoring the overall network connectivity is very crucial. In WSNs, there are three approaches for restoring connectivity: mobile node relocation, relay node placement and topology-control via transmission range adjustment. In this section we briefly discuss these approaches.

3.1.1 Mobile Node Relocation in WSNs

In the first approach, as the nodes are mobile, the main idea is to reposition the existing alive nodes to restore connectivity. The general objective in this approach is to minimize the total distance traveled by the involved nodes. One example of this method is PADRA, developed by Akkaya et al. [2]. In this approach, each node chooses one of its neighbors to be the failure handler, which will start recovery if the node dies. The restoration process only occurs if the entire network gets disconnected, in which case the closest node that can take the dead node's place is relocated to that position. Distributed Actor Recovery Algorithm (DARA) [1] is another similar approach that aims to restore connectivity using a localized and distributed algorithm. Recovery through Inward Motion (RIM) [100] is also a distributed approach that restores connectivity by relocating one of the neighbors of the dead node. While PADRA and DARA requires 2-hop neighborhood information, RIM requires 1-hop neighborhood information. However, all these approaches mainly handle single node failure and cannot be used in cases of multi-node failures or when nearest suitable node is multi-hops away.

Autonomous Repair (AuR) [36] is another approach handles multi-node failures using mobile nodes. The main idea of this approach is regrouping the healthy nodes in the center of the area by repositioning them towards one another. In this approach, all nodes assumed to be mobile and in case of a network partitioning, it is expected for so many nodes to be repositioned.

Another recent such approach is Resource Constrained Recovery (RCR) [37] proposed by Joshi and Younis. In this approach, not the alive sensor nodes, but relay nodes are repositioned to reconnect the disjoint segments of the partitioned network.

3.1.2 Relay Node Placement in WSNs

The objective in the relay node placement problem is to place minimum number of relay nodes in a region where sensor nodes are already deployed so that the resulting topology is connected and/or fault-tolerant. In the literature there are several such studies [30], [56], [41], [84], [104], [13]. There are also some recent studies like [76] to determine least number of relay nodes to maintain multi-hop paths between every pair of sensor nodes.

As stated by Liu et al [55], existing solutions to obtain fault-tolerance aim to obtain k-vertex connectivity between any two sensor nodes and achieve that using least number of relay nodes. Although such optimization is a very challenging work and proven to be NP-hard [53], it does not conform with the general objective of WSN applications. For WSN applications, general objective is to send the received data to the sink nodes. Therefore, it is more convenient to have fault-tolerant paths from sensor nodes to the sinks. Our aim differs from the existing solutions at this point. We aim to provide fault-tolerant disjoint paths between every sensor node to the set of supernodes, but not between every pair of sensor nodes.

Recently, Azharuddin and Jana [5] proposed a method to place minimum number of additional relay nodes to achieve fault-tolerance for heterogeneous WSNs. However, they do not allow sensor-to-sensor communication and thus locate relay nodes in such a way that there is at least k relay nodes within the transmission range of each sensor node. Our approach, on the other hand, allows sensor-to-sensor communication and hence potentially decreases the required number of supernodes to be placed to achieve a certain degree of fault-tolerance. GRASP-ARP [79] is another relay node placement approach that aims to deploy minimum number of relay nodes, so that in the resulting network each sensor node is connected to the sink nodes by at least k disjoint paths. In terms of the resulting network topology, GRASP-ARP has similar intentions with our approach, however being a centralized approach restricts its usage in WSN applications.

3.1.3 Topology-control via transmission range adjustment

These first two approaches, that is, mobile node relocation and relay node placement, may not be practical in real-world scenarios because sensor nodes are often deployed in remote and inhospitable regions with harsh environments that render manual node placement or relocation infeasible. Note that due to the dynamic nature of WSNs, node placement and/or relocation must be repeated periodically. In addition, these approaches require overall network information, something that is also not suitable for most real-world applications.

As a remedy to the problems discussed above, topology-control emerges as a third approach for connectivity restoration. In this approach, the topology is controlled by adjusting the sensor nodes' transmission ranges. One example of this method is RESP [88], which is an energy-aware topology-control algorithm that ensures k-edge connectivity for flat networks. The RESP algorithm assumes sensor nodes are aware of their location information via GPS or other localization techniques, and periodically updates the network topology to adapt to sensor nodes' residual battery power levels. Because of ensuring k-edge connectivity, but not k-vertex connectivity, RESP cannot keep the network connected up to $k - 1$ node failures. Another recent approach, Energy-harvesting Heterogeneous WSN (EHWSN) [99], also aims to preserve k-vertex supernode-connectivity for heterogeneous WSNs. EHWSN is a centralized approach and ignores residual battery power levels, therefore not scalable and not energy-aware.

3.2 DPV Algorithm

The aim of the DPV algorithm [6] is to minimize the total transmission power of a WSN while maintaining k-vertex disjoint paths from each sensor node to the set of supernodes. The DPV algorithm gets a k-vertex supernode-connected network topology as an input and generates a subnetwork consisting of the same set of sensors but fewer connections. The output of the DPV algorithm is a total transmission power optimized and k-vertex supernode-connected network topology. Consider the example topology given in Figure 3.1, which consists of one supernode and three sensor nodes. When the aim is to provide one-vertex supernode connectivity, DPV removes three edges and optimizes the given network topology, as in Figure 3.2. The main contribution of DPV is its efficiency in computing such network topologies. The DPV algorithm requires $O(n\Delta^2)$ message transmissions, whereas the best alternative [10] incurs $O(\Delta^5)$ messages, where n is the number of sensor nodes and Δ refers to the maximum degree of a sensor node. Note that we assume a dense network, where Δ is sufficiently large. The DPV algorithm consists of five main stages:

1. Collecting path information and calculating disjoint paths,
2. Calculating the set of required neighbors,
3. Notifying the nodes in the disjoint paths and updating the required neighbors,
4. Removing the non-required neighbors and
5. Reducing the power level to a point sufficient only to reach the farthest required neighbor.

Among these stages, first one is the most crucial one. Most of the work is done during this stage. It is initiated by the supernodes through 'Init' messages. An 'Init' message contains the ID of the supernode that created the message and can only be transmitted by a supernode. When a sensor node receives these messages it updates its local path information, where disjoint paths to the set of supernodes are stored. If there is an update in this path list, sensor node creates and transmits a 'PathInfo' message containing its local path information. Upon receiving a 'PathInfo' message, each sensor node computes the disjoint paths to the set of supernodes by using the union

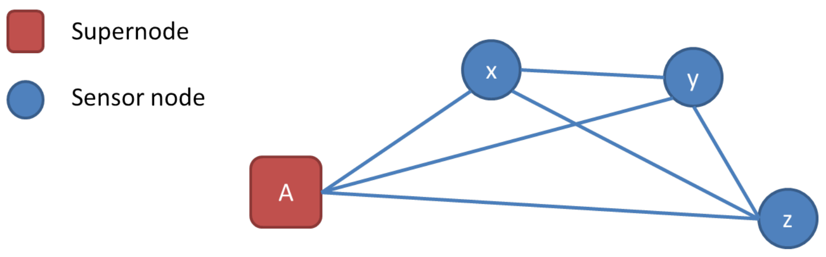


Figure 3.1: Initial network.

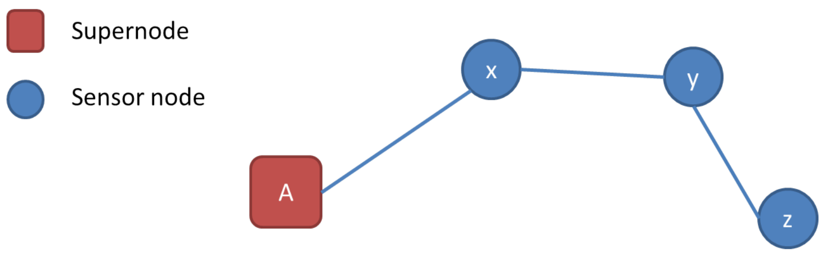


Figure 3.2: Optimized network.

of local data and the path information received from the 'PathInfo' message. If the incoming message changes the disjoint paths and decreases path costs, sensor node updates its local path data and notifies its 1-hop neighbors by transmitting a 'PathInfo' message. This process continues until no more update occurs in the network.

As a result of the first phase, each sensor node calculates *k* pairwise disjoint paths to the set of supernodes in their local path information table. During the second phase, DPV calculates its required neighbors according to the chosen disjoint paths. In the third phase, each sensor node initiates a 'Notify' for each disjoint path to inform inform the nodes on the paths regarding the path information. When a sensor node receives a 'Notify' message, it marks its 1-hop neighbors on the disjoint path as required neighbors and forwards the message to the next node on the path. In the forth stage, neighbors that are not marked as required are removed from the neigbor list and finally, transmission power is adjusted to reach the farthest remaining neighbor.

3.3 Multiple Sink Placement

Due to the scarce energy supplies of WSNs, it is necessary to design energy-efficient architectures and optimize energy consumption. In small networks, sensors can send their data directly to the sink node. In larger networks, multi-hop communication is required and most of the energy is spent for data relaying. Especially the sensors within one-hop distance from the sink have to relay the data for the other sensor nodes and most of their energy is spent on data relaying. Therefore, in order to use the energy efficiently, it is important to shorten number of hops a packet has to travel until reaching the sink and balance the relaying job among sensor nodes. Deploying multiple sinks is a method to shorten these hop counts, balance relaying job and prolong network lifetime [8, 27]. Multiple sink usage also improves various network performance including average data delivery latency [102] and system throughput [67].

WSNs experience failure problems due to various factors such as power depletion, environmental impact, radio interference, dislocation of sensor node and collision. The problem of missing sensor node and communication link errors are inevitable in WSNs. In addition to failures of sensor nodes, sinks may also fail due to different reasons such as hardware failure, software failure or intentional attacks [42, 70]. Therefore, fault-tolerance is an important design factor that should be considered during the deployment of sink nodes. Deployment solutions are mainly designed to ensure coverage and connectivity requirements of WSNs. In multiple sink placement problem, the aim is to ensure network's connectivity while keeping the maximum hop-count constraint. Together with fault-tolerance requirements, sinks are placed so that all the sensor nodes are connected to multiple sinks or connected to the set of sinks with fault-tolerant paths. In this way, multiple sink placement solutions support fault-tolerance by ensuring the existence of alternative routes to the sinks when failure occurs.

In general, finding minimum number of required sink nodes is NP-hard [73] and determining locations for these sinks is NP-complete [8]. Since optimal sink placement has proved to be NP-complete, several sub-optimal heuristics were proposed with the objective of balancing energy consumption [95, 58], reducing packet delivery latency [102] and meeting fault-tolerance requirements [80]. Xu and Liang [95] and

Oyman and Ersoy [61] aim to minimize number of deployed sink nodes, while other mentioned approaches get number of sink nodes as an input and deploy sinks into precomputed locations. Sitanayah et al. [80] and Xu and Liang [95] are the only two approaches that take maximum hop-count into consideration and Sitanayah et al. [80] is the only approach that considers fault-tolerance in the multiple sink placement problem. None of the multiple sink placement approaches consider residual energy levels of the sensor nodes and also none of the mentioned approaches is distributed.

In [66], Poe and Schmitt discuss four different sink placement strategies. These are: Random Sink Placement (RSP), Geographic Sink Placement (GSP), Intelligent Sink Placement (ISP) and Genetic Algorithm-based sink placement (GASP). Among these, GSP and GASP are most efficient strategies. Recently, Pardesi and Grover [63] improves GSP and proposed a new strategy, namely I-GSP, that divides the network in concentric circular rings around the central circular region. In [15], Dandekar and Deshmukh propose an algorithm named Optimal Multiple Sink Placement (OMSP) that divides the network into clusters for the given number of sink nodes and calculates locations for them using particle swarm optimization. Das et al. [17] proposes two algorithms, namely Candidate Location with Minimum Hop (CLMH) and Centroid of the Nodes in a Partition (CNP), and they also compare their results with Geographic Sink Placement (GSP) strategy. Das et al. assume a partitioned network, where partitions is given as an input and try to find locations for the cluster heads that will minimize transmission delay and extend network lifetime.

In this study, we present an energy-aware and distributed solution to determine sink positions and form a fault-tolerant network topology. In terms of the intended network topology, our aim differs from all existing solutions. Only one of the existing multiple sink placement solutions addresses fault-tolerance. However, that study only considers sink failures, but not sensor node failures, and provides ensures sensor nodes are connected to multiple sinks [80]. Greedy-MSP and GRASP-MSP are the two approaches mentioned in [80] to calculate multi-hop paths to multiple sinks. In this study, we consider both sink and sensor node failures and also take residual energy levels of the sensor nodes into account to calculate number of required sink nodes and their locations and achieve a more robust network topology.

3.4 Power Consumption Model

Our ADPV algorithm aims at prolonging network lifetime, and thus it should first model the amount of time until the battery powers of the sensor nodes are depleted. The ADPV algorithm uses a well-known power consumption model, proposed by Heinzelman et al. [33, 35]. This approach is based on the observation that the main factor in WSN power consumption is data communication, which consists of two factors: data transmission and data reception. In this model, the power to transmit a bit to a distance of d is

$$P_t(d) = \alpha_1 + \alpha_2 \times d^n, \tag{3.1}$$

where α_1 and α_2 are parameters that depend on the transmitter circuitry, and n is the path loss exponent for the environment, which often has a value between 2 and 4. In our power consumption model, α_1, α_2, and n are assumed to be 50 nJ/bit, $100pJ/bit/m^2$ and 2, respectively.

In our model, the energy consumption for data reception is a constant value per bit. We represent this constant with β and assume it equals 50 nJ/bit.

For our experiments, we assume all sensor nodes are sensing the environment and generating traffic at a fixed rate. We also assume that data aggregation is applied and that all nodes on a path carry the same load. Therefore, total power consumption for receiving a bit and transferring it to the next hop equals:

$$P_f(d) = \beta + \alpha_1 + \alpha_2 \times d^n. \tag{3.2}$$

If the residual battery energy level of sensor node i is denoted as e_i, then the lifetime of node i equals:

$$l_i = e_i/((r_{ri} \times \beta) + (r_{ri} + r_{gi}) \times (\alpha_1 + \alpha_2 \times d_i^n)), \tag{3.3}$$

where r_{ri} is the incoming data rate to node i, r_{gi} is the data rate generated in node i and d_i is the transmission range.

CHAPTER 4

ADAPTIVE DISJOINT PATH VECTOR ALGORITHM

In this chapter, we present our novel adaptive and distributed algorithm, ADPV, which aims to construct and maintain a k-vertex supernode-connected topology to prolong the k-vertex supernode-connected lifetime of the network. The ADPV algorithm controls the topology by adjusting the transmission ranges of sensor nodes, and to comply with real-life situations it considers node failures. The algorithm requires only one-hop neighborhood information and constructs the network topology by a series of message exchanges.

The ADPV algorithm consists of two phases: initialization and restoration. It collects necessary information and builds an initial topology during the initialization phase. Whenever a node failure breaks k-vertex supernode connectivity, ADPV restores connectivity within the restoration phase. Similar to DPV, ADPV utilizes disjoint paths and within each restoration phase each sensor node decides whether or not to change its disjoint paths. At the end of each restoration phase, sensor nodes' transmission ranges are adjusted according to the intended topology. The main differences between ADPV and DPV are as follows:

- ADPV is an adaptive approach, adapting to node failures and remaining energy levels, whereas DPV is a static one.
- ADPV considers residual battery power levels of sensor nodes, therefore it is an energy-aware solution. DPV on the other hand ignores sensor nodes' remaining energy levels.
- ADPV balances energy consumption and optimizes the lifetime of disjoint

paths, as opposed to DPV, which optimizes the total transmission power of sensor nodes.

- ADPV significantly prolongs both one-vertex and *k*-vertex supernode-connected lifetimes of the network with its solutions for restoration path selection, *k*-vertex supernode-connectivity verification and connectivity restoration.

4.1 Network Model

Consider a mission critical border surveillance system that is integrated with a two-tiered heterogeneous wireless sensor network. In this network, there are supernodes located on each tower and regular sensor nodes that are uniformly distributed into the target area as shown in Figure 4.1. In this network, sensor nodes are responsible for detecting potential intrusion activities and inform the towers by forwarding data to the supernodes located at those towers. Since it is common to lose some sensor nodes because of energy depletion, harsh environmental conditions or hostile activities of intruders, it is desired for every sensor node to have more than a certain number of independent paths to the supernodes. In the figure, we can see a soldier crossing the border, and a sensor node close-by informs some towers via three disjoint paths.

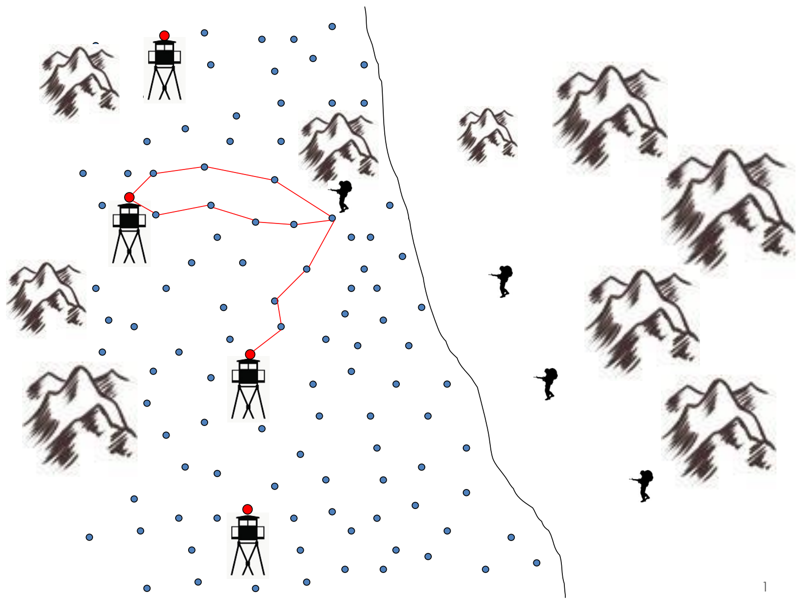


Figure 4.1: Sample scenario.

This network model is first described in [10], and also used by the DPV algorithm. In this model, the network consists of M supernodes that are deployed at known locations and N sensor nodes that are randomly distributed in the 2D plane so that $M << N$. We assume the supernodes have transmission ranges long enough to communicate with the base station or any other supernode in the network. Therefore, we do not model and are not concerned with supernode-to-supernode communication. We are only interested in sensor-to-sensor and sensor-to-supernode communication.

We represent the initial network topology with an undirected weighted graph $G = (V, E)$, where V is the set of nodes and $E = \{v_i, v_j \mid dist(v_i, v_j) < R_{max}\}$ is the set of edges; $dist(v_i, v_j)$ defines the distance between nodes v_i and v_j.

4.2 Problem Definition

We first give the formal definition of k-vertex supernode connectivity.

Definition 4.2.1 (k-vertex supernode connectivity [10]) *An heterogeneous WSN is said to be k-vertex supernode-connected if removal of any $k - 1$ sensor nodes does not disconnect any sensor node from all the supernode(s), that is, each sensor node is still connected to some supernode(s).* ■

Initially we are given a k-vertex supernode-connected network with M supernodes and N sensor nodes, where the sensor node transmission range can be adjusted up to a predefined constant R_{max}. As we model node failures, the number of active sensor nodes decreases during the network lifetime. We use N_t to denote the set of active sensor nodes at time t, where time is represented by discrete time intervals. Our problem is to determine the transmission ranges of all active sensor nodes at any time, such that the resulting topology is still k-vertex supernode-connected, so that network lifetime can be improved. Now, we formally state the problem of maximizing fault-tolerant lifetime.

Definition 4.2.2 (Fault-tolerant lifetime maximization) *Given a k-vertex supernode-connected WSN $G = (V, E)$ with a set $M \subset V$ of supernode vertices and a set $N_t \subset V$*

of active sensor node vertices, such that $M \cap N_t = \emptyset$, find a set of edges $F \subset E$ such that $G(V, E - F)$ is k-vertex supernode-connected and $\sum_{i=1}^{|N_t|} l_i$ is maximized, where l_i is the lifetime of the minimum lifetime path among the disjoint paths of $v \in N_t$. ■

4.3 Residual Battery Power Level-Aware Disjoint Path Selection

The ADPV algorithm adapts the network topology dynamically during network operation by adjusting the sensor nodes' transmission ranges according to residual energy levels. For instance, if a node has low remaining energy, it should choose closer neighbors; otherwise, it may choose farther ones. In this way, we attain a fair distribution of total residual energy among sensor nodes.

The DPV algorithm is not an energy-aware solution, and ignores sensor nodes' residual energy levels. This design may cause early battery depletion, since a node with low residual energy may be assigned to a high transmission power range. The ADPV algorithm, on the other hand, takes residual energy levels into consideration when selecting disjoint paths. Estimating the lifetime of each sensor node on a path lies at the core of our approach. The motivation behind this method is that a chain is only as strong as its weakest link, and thus a path survives only as long as all nodes survive in the path. Therefore, the shortest node lifetime on the path determines the lifetime of the path. The ADPV algorithm chooses a set of disjoint paths such that the minimum lifetime of those paths is maximized.

We formally define the lifetime of a path as follows: Let a path P consists of nodes $n_0, n_1, .., n_l$, in which n_0 is the starting sensor node and n_l is a supernode. Let b_i denote the residual energy level of sensor node n_i and d_i denote the distance between n_i and n_{i+1} for each $0 \leq i < l$. Then, the lifetime of P is defined as:

$$\text{Lifetime}(P) = \min_{0 \leq i < l} \{ b_i / (\beta + \alpha_1 + \alpha_2 \times d_i^n) \},$$

where β, α_1, α_2 and n are the constant parameters of power consumption, defined in Section 3.4.

4.4 Initialization Phase

This section describes our proposed approach for selecting alternative routes in the initialization phase of the algorithm, where those routes are to be used to restore connectivity during restoration phases. In ADPV, each sensor node keeps alternative routes, here referred to as restoration paths, that start with that node.

The primary goal is to consume the minimum possible resources while attaining high-quality restoration paths. The resources include memory, CPU, and network. Regarding memory, for instance, if all possible paths from sensor nodes to supernodes were held, the memory requirement would be intractable. In [85], Valiant discusses the average number of paths from a node to a given set of nodes. In terms of CPU, Bagci et al. [6] show that the complexity of selecting k disjoint paths from a pool of p alternatives is O(p^k). Therefore, with a higher number of restoration paths of size r, it takes longer to compute a disjoint path set of size k during each restoration phase. As for the network, which is last but not least, we aim to communicate using minimum number of messages. Each restoration path incurs communication between its nodes in order to update its lifetime. As a result, we should maintain a very restricted set of restoration paths for the sake of network performance, but at the same time, the amount of those paths should be high enough to restore connectivity whenever needed.

To overcome these restrictions and efficiently construct restoration paths, ADPV employs a well-known method, called maximum set packing (MSP) [25]. This method is the optimization version of the set packing (SP) problem and asks for the maximum number of pairwise disjoint sets among a family of sets. More formally, for a given universe $\mathcal{U}$ and a family $\mathcal{S}$ of subsets of $\mathcal{U}$, MSP is a subfamily $C \subseteq \mathcal{S}$ of sets such that all sets in C are pairwise disjoint, and C uses as many sets as possible, so that the size of the packing $||C||$ is maximum. Maximum set packing is NP-hard [40] and cannot be approximated within any constant factor [31].

For example, consider the universe $\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and a family $\mathcal{S} = \{S_a, S_b, S_c, S_d\}$, consisting of the subsets shown in Figure 4.2:

- $S_a = \{1, 2, 3, 4\}$

- $S_b = \{4, 5, 6\}$
- $S_c = \{5, 6, 7, 8\}$
- $S_d = \{9, 10\}$

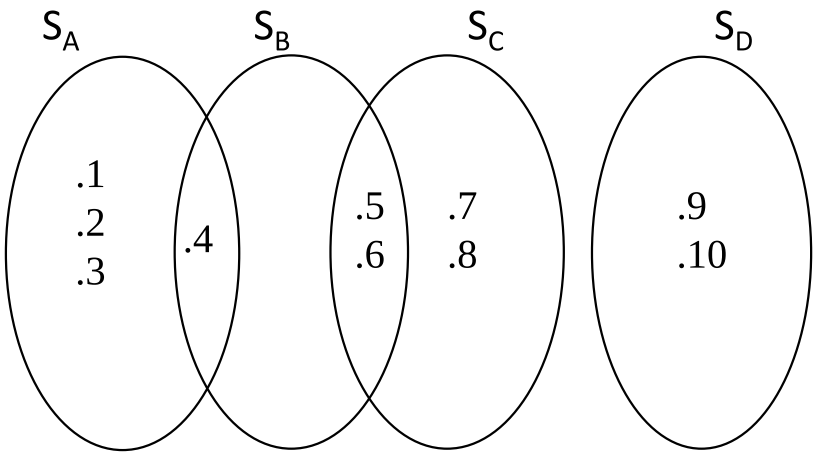


Figure 4.2: Subsets of family $\mathcal{S}$.

Pairwise disjoincy of these subsets is as follows:

- S_a is pairwise disjoint with S_c and S_d
- S_b is pairwise disjoint with S_d
- S_c is pairwise disjoint with S_a and S_d
- S_d is pairwise disjoint with S_a and S_b.

For this example, since S_b is not pairwise disjoint with S_a and S_c, any set packing C that contains S_b can at most has 2 elements. Excluding S_b, set $\{S_a, S_c, S_d\}$ contains 3 pairwise disjoint elements, therefore is a MSP.

There is a well-known greedy heuristic, shown in Algorithm 4.4.1, to solve the MSP problem and it runs in polynomial time. We employ this heuristic to construct restoration paths. At the beginning, we have a pool of candidate paths of a relatively large size. The heuristic performs with many iterations, where each iteration selects the most diverse path from the pool. We use the term *diverse* as being disjoint with others, that is, the one that is disjoint to the largest number of paths among others in the pool. We add the selected path into the restoration path set and remove all the paths from the pool that intersect with the selected path. The iterations continue until the pool becomes empty or the number of restoration paths reach a predefined threshold. Since the initial sensor node and the destination supernode do not violate disjoincy, ADPV represents each path by the set of its intermediate sensor nodes.

Algorithm 4.4.1 Maximum Set Packing (MSP)

```
Input: $S$
Output: $M$
 1: $M \leftarrow \emptyset$;
 2: while $S \neq \emptyset$ do
 3:     $m \leftarrow$ MinIntersectingPath($S$);
 4:     $M \leftarrow M \bigcup m$;
 5:     for all Path $p \in S$ do
 6:         if $p \bigcap m \neq \emptyset$ then
 7:             $S \leftarrow S - p$;
 8:         end if
 9:     end for
10: end while
```

Algorithm 4.4.2 shows path-information-collection and restoration-path-selection procedures. The variables used in the pseudo codes are defined in Table 4.1. In Algorithm 4.4.2, each sensor node maintains a local path set along with disjoint and restoration path sets. As an input, the algorithm takes a ‘PathInfo’ message that contains the local path set of the sender node and generates two outputs, which are the disjoint path and restoration path sets of size k and a relatively large size, respectively. Local paths are logical paths that are used for informing neighbor nodes about the paths they can use over the sender node. Therefore, local paths have a very critical role in determining disjoint and restoration path sets and need to be selected very carefully. When a sensor node receives a ‘PathInfo’ message containing a local path set, it first calculates the union of the sender’s and receiver’s local path sets. It then executes the MSP procedure on this union to eliminate paths that have too many sensor nodes in common. The procedure then determines a candidate local path set T' as the first L minimum-cost path of the remaining set. While doing so, the procedure also updates the restoration paths by executing the MSP procedure on the union of local sets and the current restoration path set.

Algorithm 4.4.2 Path Information Collection in ADPV

```
Input: $I, L, k$
Output: $D, R$
 1: $T \leftarrow \emptyset$;
 2: $R \leftarrow \emptyset$;
 3: for all received PathInfo message $I$ do
 4:     $D \leftarrow$ MinDisSet($T, k$);
 5:     $c \leftarrow$ Cost($D$);
 6:     $U \leftarrow I.T \cup T$;
 7:     $R \leftarrow$ MaxSetPacking($R \cup U$); (Algorithm 4.4.1)
 8:     $U' \leftarrow$ MaxSetPacking($U$);
 9:     Sort($U'$);
10:     $T' \leftarrow \{p_i \in U' \mid i <= L\}$;
11:     $D' \leftarrow$ MinDisSet($T', k$);
12:     $c' \leftarrow$ Cost($D'$);
13:     if $c' < c$ then
14:         $T \leftarrow T'$;
15:         Transmit PathInfo($T$);
16:     end if
17: end for
```

Using the candidate local path set, the set of disjoint paths with minimum cost is calculated using Algorithm 4.4.3. In this algorithm, all disjoint subsets with k elements are traversed and the one with the minimum cost is selected. If the minimum-cost disjoint path set has a smaller cost than the current disjoint path set, both the disjoint and the local paths are updated and a ‘PathInfo’ message containing the new local path set is transmitted to the set of neighbors. This process continues until there are no more updates in the disjoint path sets.

After determining the disjoint paths, each sensor node determines its required neighbors, which include the neighbors that disjoint paths use the edges between. After determining the required neighbors, each node adjusts its transmission power to reach its farthest neighbor according to the resulting topology.

4.5 Connectivity Restoration Phase

We start the connectivity restoration procedure only when k-vertex supernode connectivity is broken due to node failure. Thus, the first step after a node failure is to check whether the network is still k-vertex supernode-connected or not. As this is a costly

Table 4.1: ADPV notations.

I	Received PathInfo message
L	Maximum number of paths to be stored
k	Disjoint connectivity degree
R	Set of restoration paths
T and T'	Set of local paths
D and D'	Set of disjoint paths
c and c'	Cost of disjoint paths, which equals the minimum lifetime of the disjoint paths
U	Union of two path sets
S	Set of paths
M	Set of paths in MSP
m, p, r	Variables referencing paths
sr	Supernode ratio
n	Total number of sensors
Δ	Maximum degree of a node
r	Amount of a node's restoration paths
l	Average path length in the restoration set
$n_0, n_1, .., n_i$	Number of remaining sensor nodes after each restoration phase

Algorithm 4.4.3 Finding Disjoint Paths to Supernodes (MinDisSet)

```
Input: $T$ and $k$
Output: $D$
 1: $D \leftarrow \emptyset$;
 2: if $|T| > k$ then
 3:     $Q \leftarrow \{ q \subset T \mid |q| = k \}$;
 4:     $c \leftarrow \infty$;
 5:     $q_{min} \leftarrow \emptyset$;
 6:     for all $q \in Q$ do
 7:         if $q$ consists of disjoint paths then
 8:             if Cost($q$) $< c$ then
 9:                 $c \leftarrow$ Cost($q$);
10:                 $q_{min} \leftarrow q$;
11:             end if
12:         end if
13:     end for
14:     $D \leftarrow q_{min}$;
15: end if
```

operation [23], ADPV employs a simple distributed greedy heuristic with no false positives. That is, if ADPV postulates the network is k-vertex supernode-connected, then the network is definitely connected. However, the network can still be connected even if ADPV claims it is not. Therefore, ADPV ensures strong k-vertex supernode connectivity.

When a node failure occurs, ADPV ensures all the node's neighbors initiate a failure message to inform others about the failure. Upon receiving a failure message, a sensor node removes all paths including the failed node from its restoration set. Since frequent transmission power adjustment is difficult to realize in practice, we employ periodical transmission power control, and during each period we check whether any failed nodes exist on any of the disjoint paths. If a failed node disconnects a disjoint path, the restoration process takes place. Note that this event does not necessarily imply k-vertex supernode disconnectivity, yet because ADPV takes early action it never allows the connectivity to break. After deciding k-vertex supernode connectivity must be restored, ADPV applies a two-step process: updating the lifetimes of the restoration paths and computing minimum-cost disjoint paths from the restoration set.

In the first step, path lifetimes in the restoration set are updated via messages transmitted along the path from the source node to the destination supernode. Each node redirects a received message to the next hop in the path and returns a message that contains updated lifetime information of the sensor nodes back along that path. In the second step, minimum-cost disjoint paths are computed using the previously discussed disjoint-path-selection algorithm, Algorithm 4.4.3. An overview of the connectivity-checking and connectivity-restoration procedures are given in Algorithm 4.5.1.

Algorithm 4.5.1 Connectivity Restoration in ADPV

```
Input: $k, R, D$
Output: $D$
 1: $FailedNodes \leftarrow \emptyset$;
 2: for all received node failure message $\delta$ do
 3:   for all Path $r \in R$ do
 4:     if $r$ contains $\delta$.FailedNode then
 5:       $R \leftarrow R - r$;
 6:     end if
 7:   end for
 8:   $FailedNodes \leftarrow FailedNodes \cup \delta.FailedNode$
 9:   if certain time elapsed since last period then
10:     for all Path $p \in D$ do
11:       if $(p \cap FailedNodes) \neq \emptyset$ then
12:         UpdateCosts($R$);
13:         $D \leftarrow$ MinDisSet($R, k$);
14:         break;
15:       end if
16:     end for
17:     $FailedNodes \leftarrow \emptyset$;
18:   end if
19: end for
```

For instance, continuing from the example given in Section 3.2, for $k = 2$, ADPV optimizes the topology shown in Figure 3.1, as in Figure 4.3(a). In this topology, all initial energy levels are equal. Assuming the data generation rate is uniform for all nodes, the power consumption of nodes x, y and z are 1.2, 2 and 1, respectively. With this power consumption, node y dies first (100/2=50 seconds later), both node x and node z lose one of their disjoint paths and the network becomes one-vertex, but not two-vertex, supernode-connected. The ADPV algorithm restores connectivity, as in Figure 4.3(b), by adjusting the transmission range of z, which introduces a link from node z to supernode A. Because of its increasing power consumption, node z happens to be the second dying node (50/5=10 seconds later) and thus node x loses two-vertex supernode connectivity once more. However, because it has no alternative routes, it adjusts its transmission power again and works as a connected node, as in

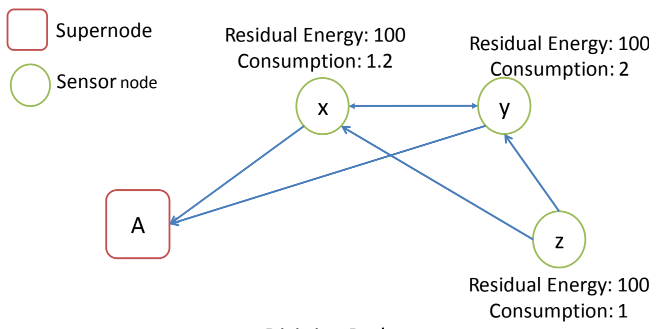
Figure 4.3(c), for the rest of its life (28/1=28 more seconds). For this network, the two-vertex supernode-connected lifetime is broken when node z dies. Therefore, the two-vertex supernode-connected lifetime of this network equals 60 seconds and the one-vertex supernode-connected lifetime equals 88 seconds.

Lemma 4.5.1 *The connectivity restoration process of ADPV ensures k-vertex supernode connectivity.*

Proof. By definition, the network gets k-vertex supernode-connected if each sensor in the network is connected to at least one supernode with k-vertex disjoint paths. This translates into the disjoint path set of each sensor node of being size k, and if there exist more than k paths in the restoration set, ADPV chooses a disjoint set and ensures k-vertex supernode connectivity. ■

Lemma 4.5.2 *In the restoration path set, there are at most Δ paths, where Δ is the maximum degree of a sensor node.*

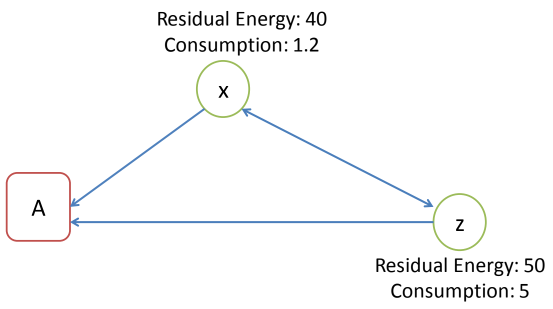
Proof. We are going to prove this by contradiction. As discussed in Section 4.5, each sensor node keeps a maximum set pack of some size in their restoration sets, so that each path in the set is pairwise disjoint with the others. Let Δ denote the maximum degree of a node and assume there exists a node, say node i, that has more than Δ paths in its restoration set. Since there are more than Δ paths that are using at most Δ neighbors, according to the pigeonhole principle, there exist two restoration paths that use the same neighbor. Let Figure 4.4 represent node i and its one-hop neighbors. If the neighbor that two paths have in common is a supernode, then node i will have exactly the same two paths in its restoration set, which is not possible, because the MSP procedure calculates the union of the selected paths to guarantee diversity. If that neighbor is a sensor node, those paths will not be disjoint, which violates the MSP definition. Therefore, no neighbor, neither sensor node nor supernode, can have two paths in common, and the number of elements in the restoration set cannot exceed Δ. ■



Disjoint Paths

x	y	z
x->A	y->A	z->y->A
x->y->A	y->x->A	z->x->A

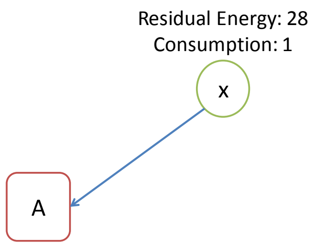
(a)



Disjoint Paths

x	z
x->A	z->A
x->z->A	z->x->A

(b)



Disjoint Paths

x
x->A

(c)

Figure 4.3: Sample connectivity restoration for $k = 2$.

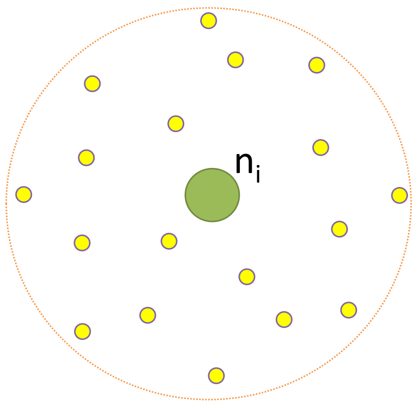


Figure 4.4: One-hop neighbors of node *i*.

4.6 Running Time Analysis

We compute the lifetime of a restoration path via messages transmitted along the path from the source node to the destination supernode. Each node redirects a received message to the next hop in the path and returns a message that contains the sensors' updated lifetime information back along the same path. Therefore, for each restoration path, the number of messages equals two times the length of the path. We assume that path length is bounded by a constant, say l, following previous studies [6]. Notice that the number of restoration paths is less than or equal to Δ, where Δ is the maximum degree of a node. Then, there are at most $l \times \Delta$ messages in total, and thus, the message complexity is $O(\Delta)$ at each connectivity-restoration phase. In the worst case, for each sensor node, connectivity restoration is carried out for $O(\Delta)$ times, as the restoration path set embodies at most $l \times \Delta$ nodes. Therefore, at each sensor node, total message complexity becomes $O(\Delta^2)$ for connectivity restoration. For path information collection, ADPV has the message complexity of $O(n\Delta)$, which also equals that of DPV [6]. Therefore, the total message complexity becomes $O(\Delta^2) + O(n\Delta) = O(n\Delta)$.

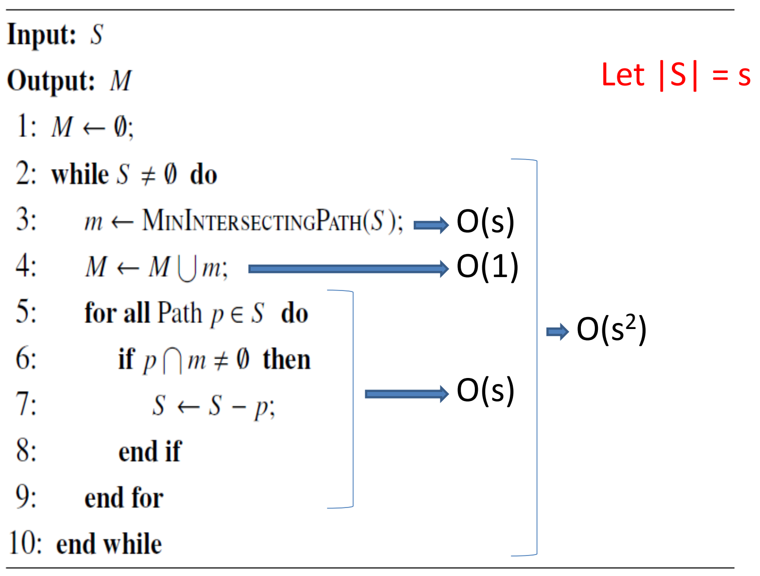
The ADPV algorithm consumes computational power in the initialization phase for disjoint and restoration path construction and during the connectivity restoration phase for determining new disjoint paths from the restoration set. During the initialization phase, when sensor nodes receive a 'PathInfo' message, they calculate the union of the local path information and the received paths in the incoming message. The running time complexity of this step depends on the number of paths (p) in the local path information table. In ADPV, since the maximum number of paths that can be stored in a sensor node's path information table is set to a constant value, both calculating the union of the two path information tables and sorting the paths according to their costs take constant time.

In the initialization phase, there are two more procedures that consumes processing power: maximum set packing and disjoint-path-selection algorithms. The greedy heuristic for MSP, shown in Algorithm 4.4.1, is used twice: once for constructing the restoration path and again for selecting the local path information table. As discussed in the second lemma, the number of restoration paths is limited by the maximum degree of node (Δ), and the number of paths in the local path information table is a constant (l). Therefore, the MSP algorithm consists of numerous iterations, each consisting of two steps: i) selecting the minimum intersecting path and ii) removing the paths that intersect with it. In the latter step, the algorithm traverses all path pairs and determines the minimum intersecting one. The activity of removing the intersecting paths also traverses the set once more. Considering set size is represented by s, the running time complexity of the MSP algorithm equals $O(s^2 + s)$. Step-by-step details of running time complexity of MSP algorithm is shown in Figure 4.5. Therefore, the MSP running time complexity in each step is $O(\Delta^2 + \Delta + l^2 + l)$, which can be reduced to $O(\Delta^2)$.

To calculate the minimum disjoint set, Algorithm 4.4.3 enumerates all subsets of size k and finds the set with the minimum cost. Enumerating all these subsets takes $O(p^k)$, where p represents the number of paths in the given set. Since the input of the minimum disjoint set procedure is the local path information table, which has a constant number of elements, the running time complexity of the minimum disjoint set calculation is also a constant.

Considering that the message transmission complexity of ADPV is $O(n\Delta)$ and the dominating step (MSP) is executed once for every incoming message, the running time complexity of the total initialization phase is $O(n\Delta^3)$.

For the restoration phases, as discussed above, the maximum number of restoration



```
Input: $S$
Output: $M$
1: $M \leftarrow \emptyset$;
2: while $S \neq \emptyset$ do
3:     $m \leftarrow$ MinIntersectingPath($S$);
4:     $M \leftarrow M \bigcup m$;
5:     for all Path $p \in S$ do
6:         if $p \bigcap m \neq \emptyset$ then
7:             $S \leftarrow S - p$;
8:         end if
9:     end for
10: end while
```

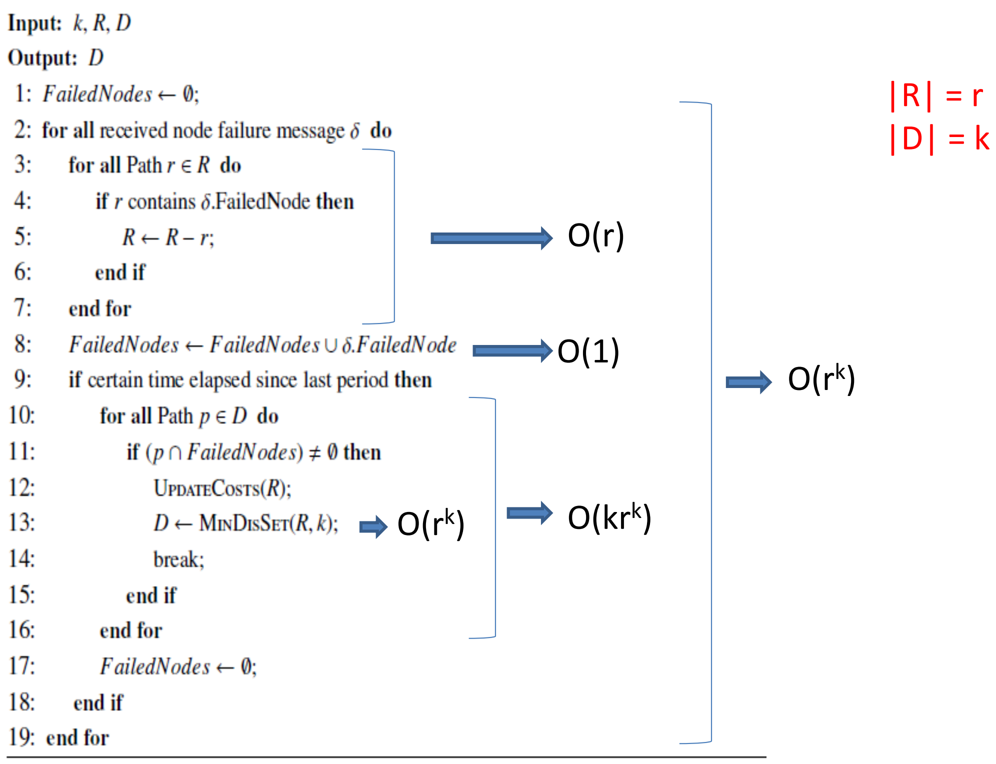
Figure 4.5: Running time complexity of MSP algorithm.

phases a node can execute is O($l\Delta$), and in each phase there are two operations: updating path costs, which only uses message transmissions, and calculating the minimum disjoint set from the restoration set. Details of the running time complexity of each restoration phase is given in Figure 4.6. Since the maximum number of elements in the restoration set is Δ, the running time complexity of one time execution of restoration phase for determining the minimum-cost disjoint paths from the restoration set will be O(Δ^k), and the total running time complexity of the restoration phases will be O(Δ^{k+1}).

Since $n >> \Delta$, and the commonly accepted values of k are 2 and 3 [10], the ADPV running time complexity equals O($n\Delta^3$).

4.7 Expected Number of Restorations in ADPV

In this section we discuss theoretical expectations resulting from the ADPV algorithm and analyze how many times ADPV can restore k-vertex supernode connectivity for a given node. Since ADPV can restore such connectivity when there are at least k paths in the restoration set, we will determine the expected number of node failures before



```
Input: $k, R, D$
Output: $D$
1: $FailedNodes \leftarrow \emptyset$;
2: for all received node failure message $\delta$ do
3:     for all Path $r \in R$ do
4:         if $r$ contains $\delta$.FailedNode then
5:             $R \leftarrow R - r$;
6:         end if
7:     end for
8:     $FailedNodes \leftarrow FailedNodes \cup \delta.FailedNode$
9:     if certain time elapsed since last period then
10:        for all Path $p \in D$ do
11:            if $(p \cap FailedNodes) \neq \emptyset$ then
12:                UpdateCosts($R$);
13:                $D \leftarrow$ MinDisSet($R, k$);
14:                break;
15:            end if
16:        end for
17:        $FailedNodes \leftarrow \emptyset$;
18:    end if
19: end for
```

Figure 4.6: Running time complexity of restoration phase.

a node cannot restore its connectivity. Let n denote the number of sensor nodes in the network and assume the sensor node batteries deplete uniformly in any order with the same probability ρ. The parameters used in this section are given in Table 4.1.

Considering that the number of sensor nodes in the restoration set equals $r \times l$, the expected number of sensor nodes that die before one of these $r \times l$ sensor nodes dies equals:

$$\frac{n}{r \times l}. \tag{4.1}$$

For example, if there are 100 nodes in the entire network and 20 take part in the restoration set, then the expected number of node failures before one of the nodes in the restoration set fails equals five. When a node on a path dies, then that path will no longer be valid and therefore will be removed from the restoration sets available. As a result, with a node failure, the number of restoration paths will diminish by one. Therefore, when the first node on a restoration set dies, $r - 1$ paths, which consist of $(r - 1) \times l$ sensor nodes, will remain. At the same time, the number of remaining sensor nodes in the entire network will equal:

$$n - \frac{n}{r \times l}. \tag{4.2}$$

Continuing from the previous example, $100 - 5 = 95$ sensor nodes will remain in the entire network after the first node in the restoration set dies. The remaining sensor nodes after the i*th* restoration path removal can be generalized as follows:

$$n_{i+1} = n_i - \frac{n_i}{(r - i) \times l}, \tag{4.3}$$

which also equals:

$$n_{i+1} = n_i \times (1 - \frac{1}{(r - i) \times l}) \tag{4.4}$$

and which can also be written as a product of:

$$n_{i+1} = n \times \prod_{j=0}^{i} (1 - \frac{1}{(r-j) \times l}). \tag{4.5}$$

Since ADPV can restore k-vertex supernode connectivity when there are at least k paths in the restoration set, the number of sensor nodes when k-vertex supernode connectivity of the given node cannot be restored equals n_{r-k+1} and can be calculated as:

$$n_{r-k+1} = n \times \prod_{j=0}^{r-k} (1 - \frac{1}{(r-j) \times l}). \tag{4.6}$$

Then by changing the parameter to $t = r - j$,

$$n_{r-k+1} = n \times \prod_{t=k}^{r} (1 - \frac{1}{t \times l}). \tag{4.7}$$

According to the above formula, the number of successful restorations will be proportional to the sensor node count. Also, with the increasing average path length, the number of remaining sensors increases, which in turn decreases the possibility of successful restorations. Therefore, choosing paths with smaller path lengths may be preferable.

CHAPTER 5

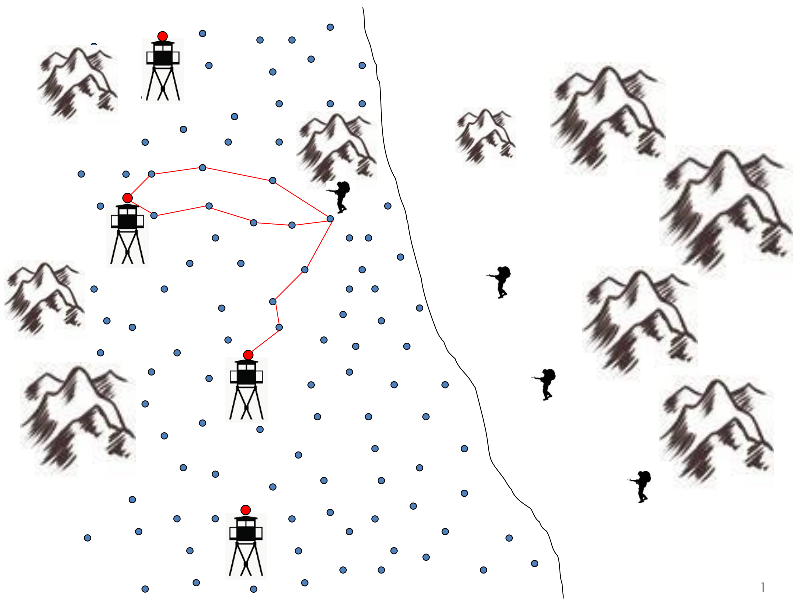
MINIMUM SUPERNODE DISJOINT PATH VECTOR ALGORITHM

In this chapter, we present our novel energy-aware and distributed algorithm, MSDPV, which aims to determine minimum number of supernodes and their possible locations to construct a k-vertex supernode-connected network topology. In this chapter, we first discuss the related network model and then describe our algorithm in detail.

5.1 Network Model

Consider a mission critical border surveillance system that is integrated with a two-tiered heterogeneous WSN. In this network, sensor nodes are randomly and uniformly deployed on the border line and initially there are no supernodes as shown in Figure 5.1(a). After MSDPV algorithm determines supernode locations, supernodes that are located on the steerable artillery units are deployed as shown in Figure 5.1(b). When sensor nodes sense a potential intrusion activity, they inform the artillery units by forwarding the data to the supernodes and artillery units decide whether to attack or not. In this scenario, fault-tolerance is achieved by forwarding the sensed data to the set of supernodes using three disjoint paths as illustrated in Figure 5.1(b).

We represent the initial network topology with an undirected weighted graph $G = (V, E)$, where V is the set of nodes and $E = \{v_i, v_j \mid dist(v_i, v_j) < R_{max}\}$ is the set of edges, where $dist(v_i, v_j)$ defines the distance between nodes v_i and v_j and R_{max} is the maximum transmission range.


(a) Initial network

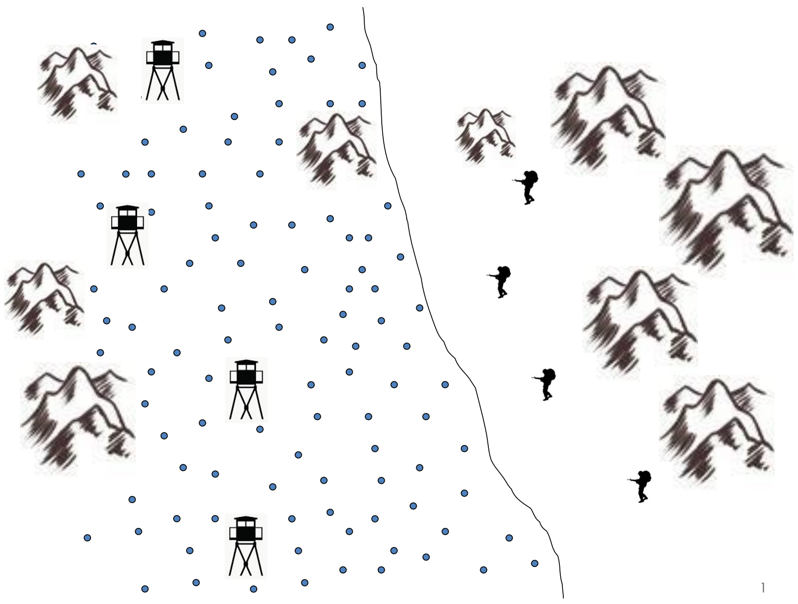

(b) After supernode deployment

Figure 5.1: Sample scenario.

5.2 Collecting Path Information and Determining Preferred Paths

The MSDPV algorithm involves two steps. In the first step, the algorithm treats all sensor nodes as potential supernodes and computes paths from sensor nodes to the supernodes. In this way, sensor nodes choose a set of other sensor nodes that they prefer to be a supernode. The second step chooses minimum number of sensor nodes to be supernodes, so that network gets k-vertex supernode connected.

MSDPV is a distributed algorithm that determines possible locations for the supernodes by message transmissions among sensor nodes and requires only one-hop neighborhood information. MSDPV is also an energy-aware solution that considers residual battery power levels of the sensor nodes. The MSDPV algorithm consists of two main steps:

1. Collecting path information and determining preferred paths,
2. Determining supernode locations.

During the first step, MSDPV uses a modified version of ADPV algorithm. Unlike ADPV, MSDPV's aim is not to determine transmission powers of the sensor nodes, but to determine possible locations for the supernodes. For this aim, MSDPV has a simple, yet effective modification over ADPV algorithm. During the first phase, MSDPV determines preferred paths for each sensor node, which are used as alternative paths for the aim of restoring connectivity in the ADPV algorithm described in Chapter 4.

Initiation method of MSDPV also differs from the ADPV algorithm. ADPV algorithm starts with the 'Init' messages initiated by the supernodes and, in the case of MSDPV, the aim is to determine supernode locations and initially there are no supernodes. MSDPV algorithm is initiated by the 'Init' messages transmitted by the ordinary sensor nodes. In this way, every sensor node behaves like a supernode and lets sensor nodes compute disjoint paths destined to itself. Since ADPV determines minimum cost disjoint paths to the set of supernodes, MSDPV determines minimum cost disjoint paths to a subset of sensor nodes. In this way, ignoring the intermediate sensor nodes and just considering the destination nodes, every sensor node chooses a

Table 5.1: MSDPV notations.

N	Set of sensor nodes
P	Union of preferred paths for all sensor nodes
T	Destination node, source node pairs of preferred paths
k	Disjoint connectivity degree
M	Set of chosen supernodes
U	List of uncovered sensor nodes
s	Chosen sensor node
m, n, x, y	Variables referencing sensor nodes

subset of sensor nodes that it prefers to be a supernode.

5.3 Determining Supernode Locations

Second phase makes use of the preferred paths calculated by each sensor node during the first phase and determine a subset of sensor nodes that will cover all sensor nodes. Aside from preferred paths, this phase requires two more inputs, which are list of sensor nodes and expected fault-tolerance degree (k). It has a single output, which is the locations for the supernodes. Since only source and destination nodes used in the second phase, as a first step, MSDPV ignores all the intermediate nodes in the preferred paths. Considering all the paths in the preferred list are pairwise disjoint, selecting any k elements from this set and choosing the destination nodes as supernodes will enable a k-vertex supernode connectivity.

Algorithm 5.3.1 shows the details of the second phase. The variables used in the pseudo code are defined in Table 5.1. Since MSDPV aims to determine set of destination nodes that will cover all the sensor nodes at least for k times, MSDPV uses a list of uncovered elements that initially contains every node for k times. To determine the smallest subset of sensor nodes that will cover all sensor nodes, MSDPV uses an optimization based on the well-known set-cover problem. Set-cover problem is one of the Karp's 21 NP-complete problems [40] and optimization version of this prob-

Algorithm 5.3.1 Determining Supernode Locations (DSL)

```
Input: $N$ and $P$ and $k$
Output: $M$
1: $M \leftarrow \emptyset$;
2: $T \leftarrow \{\ m, n \mid\ \exists\ p \subset P\ where\ m = p.Dst,\ n = p.Src\ \}$;
3: $U \leftarrow \emptyset$;
4: $i \leftarrow 0$;
5: while $i \leq k$ do
6:     $U \leftarrow U \bigcup N$;
7:     $i \leftarrow i + 1$;
8: end while
9: while $U \neq \emptyset$ do
10:    $s \leftarrow$ MaxIntersection($U$, $T$);
11:    $M \leftarrow M \bigcup \{s\}$;
12:    $U \leftarrow U - \{\ x \mid x \in N,\ (s, x) \in T\ \}$;
13:    $T \leftarrow T - \{\ (s, y) \mid y \in N,\ (s, y) \in T\ \}$;
14: end while
```

lem, which is the one required in our case, is an NP-hard problem [45]. Therefore, we use a well-known greedy heuristic to determine the smallest subset that covers all the sensor nodes. This heuristic works in iterations and in each iteration it asks for the node that contains the largest number of uncovered elements. It adds the chosen node to the result and removes all the nodes it covers from the uncovered nodes list. The iterations terminate when no elements remain in the uncovered list. This greedy approach guarantees coverage of all nodes and also approximately guarantees the minimum number of supernodes [81]. In the case of set-packing heuristic, which is used for constructing preferred paths, there was no guarantee for the optimality of the result, but for the set-cover heuristic there is.

For instance, consider the example given in Section 4.4 about maximum set packing. In this section for the same given sets we are going to calculate maximum set cover. For the given sets, shown in Figure5.2, universe $\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and a family $\mathcal{S} = \{S_a, S_b, S_c, S_d\}$, consisting of the subsets:

- $S_a = \{1, 2, 3, 4\}$
- $S_b = \{4, 5, 6\}$
- $S_c = \{5, 6, 7, 8\}$
- $S_d = \{9, 10\}$

When all the elements are needed to be covered once, first iteration of maximum set cover heuristic chooses one of the subsets S_a or S_c, because they have the most elements. Since S_a and S_c has the same number of elements, maximum set cover heuristic randomly chooses one of them. Let the chosen subset be S_a. After choosing S_a, the algorithm removes the elements the chosen subset covers from the other subsets. Since S_a intersects with only S_b, the intersection is removed from S_b and after the first iteration S_b has two remaining elements, which are $\{5, 6\}$. During the second iteration, since S_c has the most uncovered elements, the algorithm chooses $S_c$and removes the elements it covers from other subsets. As a result of second iteration, subset $S_b$has no elements. During the third iteration, the algorithm chooses S_d and as a result, no elements remain uncovered and the algorithm terminates with the chosen subsets S_a, S_c and S_d.

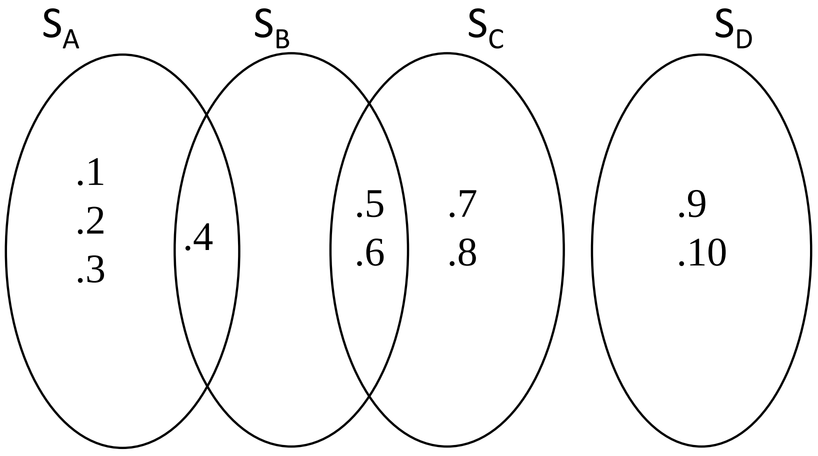
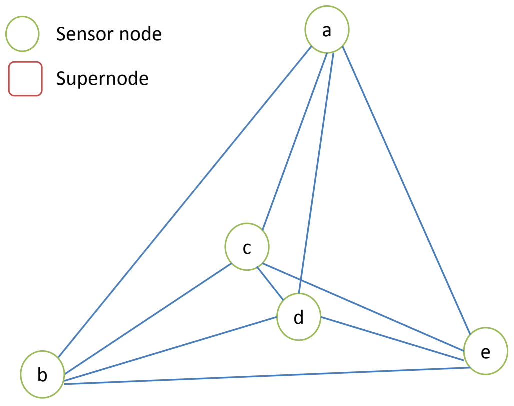


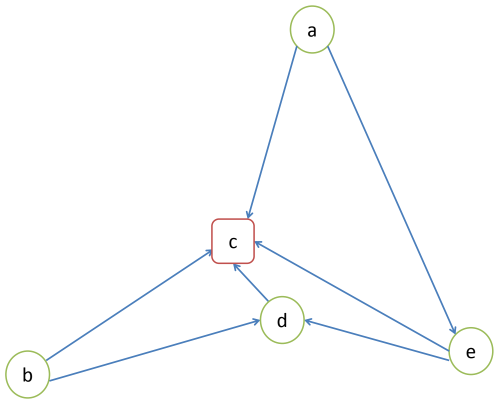
Figure 5.2: Maximum set cover example.

As an example of sample WSN, consider the sample topology given in Figure 5.3(a) with the desired connectivity degree (k) value of two. In this topology there are five sensor nodes, so that all of them are initially pairwise connected. During the first step all the sensor nodes behave like they also have supernode capabilities and asks sensor nodes to calculate disjoint paths destined to themselves. Sensor nodes compute their preferred paths to other sensors using the initialization phase of ADPV algorithm, described in Section 4.4. According to these paths, ignoring the intermediate nodes and denoting number of occurrences with exponents, destination nodes of node a are c^2, e and d; destination nodes of node b are c^2 and d^2; destination nodes of node c are a,b,d^3 and e; destination nodes of node d are b, c^3 and e; finally destination nodes of node e are c^2 and d^2. During the second step, using the set-cover algorithm that chooses the node that covers most elements in each iteration, MSDPV algorithm chooses node c to be a supernode. In this way, all the sensor nodes in the uncovered node list are covered twice and uncovered node list becomes empty. Therefore one supernode is adequate for this sample topology when $k = 2$. Final topology, where a supernode is located at the location of node c is shown in Figure 5.3(b).

With the increasing k value, the additional load on the MSDPV algorithm does not significantly increase and MSDPV algorithm works very efficiently. Continuing from the example above, when $k = 3$, there will be no change in the first step, but uncovered node list, in the second step, will initially have three copies of each node. During the first iteration node c will be chosen and there will remain one copies of node a, b and e in the uncovered node list. In the second iteration, node d will be chosen and algorithm will terminate with the decision of placing two supernodes to the locations of node c and d.



(a) Initial topology



(b) Final topology

Figure 5.3: Example supernode location determination for $k = 2$.

CHAPTER 6

EXPERIMENTS AND RESULTS

In this chapter we report our measurements for ADPV and MSDPV algorithms and try to evaluate their success. In Section 6.1, we provide experimental results regarding lifetime and other metrics for ADPV algorithm and in Section 6.2, we provide experimental results regarding minimum number of supernodes and lifetimes for MSDPV algorithm.

6.1 Experimental Results for ADPV Algorithm

In this section we report our measurements regarding lifetime and other metrics for the DPV and ADPV algorithms and try to evaluate ADPV's success. For this evaluation, we implemented ADPV using an extended version of a custom simulator, which has also been used for evaluating the DPV algorithm. We added a time dimension and a battery model into the existing framework and thus provided an environment that could evaluate network lifetime.

6.1.1 Experimental Setup

In our experiments, we assumed that sensor nodes and supernodes are uniformly and randomly deployed in an area of 600m x 600m and that the initial maximum transmission range $R_{max}$of the sensor nodes is set at 100m. We repeated our experiments for $\{100, 150, \ldots, 500\}$ sensor node, for $k = 2, 3$ (as these are commonly accepted k values), and for a supernode ratio (sr) of 5% and 10% over the region. Finally, we as-

Table 6.1: Simulation parameters for ADPV.

Deployment Area	600m x 600m
Initial Transmission Range of Sensor Nodes: R_{max}	100m
Number of Sensor Nodes: N	$[100 \ldots 500]$
Number of Supernodes: M	5% and 10% of N
Degree of Disjoint Connectivity: k	2 and 3
Packet Loss Rate	10%

sumed a packet loss rate of 10% for each message transmission. As a result, we had 9×2×2 experimental instances, and on each we executed both algorithms 20 times and reported the averages. Our simulation parameters are summarized in Table 6.2.

6.1.2 Results

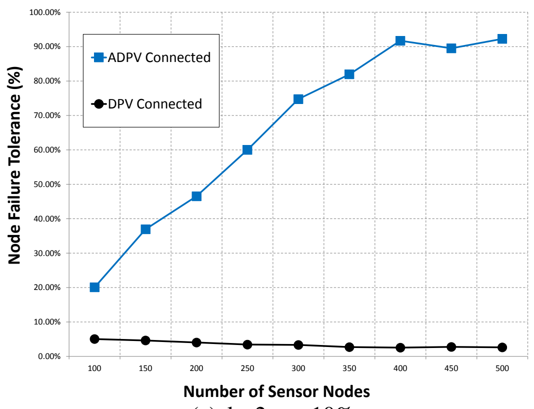
In Figure 6.2, we compare the node failure tolerance of DPV and ADPV. For each algorithm, we measure performance in terms of the fraction of dead sensor nodes when the network gets (i) supernode disconnected and (ii) k-vertex supernode disconnected. If there exists a path (single or multi-hop) between a sensor node and any one of the supernodes, then the sensor node is said to be connected. If every (alive) sensor node in the network has k disjoint paths to the set of supernodes, then the network is considered as k-vertex supernode-connected. With these measurements we determine the maximum number of node failures that can occur before supernode connectivity is broken. Here, we observe the most striking result, and at the same time, evidence of this study's motivation regarding the instability of static algorithms and effectiveness of ADPV for keeping the network supernode-connected. As seen in the Figure 6.1, even before the failure of 5% of the sensor nodes, the network's supernode connectivity is broken when we employ the DPV algorithm. Also, Figure 6.2 compares the results for 2- and 3-vertex supernode connectivity and when DPV algorithm is employed, networks k-vertex supernode-connectivity gets broken even before the failure of less than 1% of the sensor nodes. These results limit using DPV as a fault-tolerant alternative. On the other hand, as can be observed in Figures 6.1 and 6.2, ADPV successfully keeps the network supernode-connected up to failure of about 90% and

k-vertex supernode-connected up to failure of 85% of the sensor nodes.

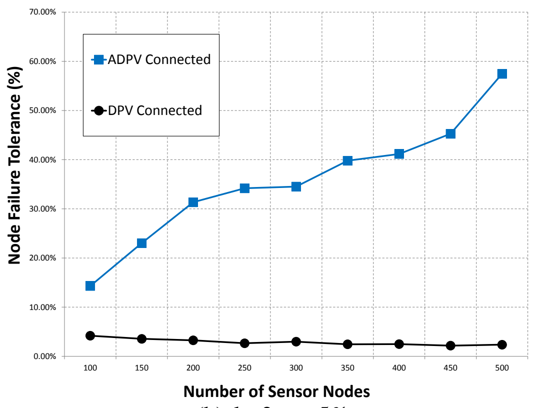
In the figure, we observe that when the network becomes denser, ADPV keeps it supernode-connectivity for longer. This result can be attributed to ADPV becoming more effective at finding alternative routes due to the increasing number of sensor nodes. For instance, in one extreme, when we examine the results of a 500-node network for $k = 2$ and $sr = 10\%$, as shown in Figure 6.1(a), we see that the network is still supernode-connected up to a failure of 95% of the sensor nodes. In the other extreme, where the number of initially deployed sensor nodes equals 100, ADPV sustains supernode connectivity up to the failure of 20% of the sensor nodes. Looking into each of the sub-figures in Figure 6.1 , when the initial number of sensor nodes is between 250 and 300, we notice that ADPV succeeds in keeping supernode connectivity even after the active sensor nodes are halved. Considering all experimental instances, on average, ADPV maintains supernode connectivity up to a failure of 52% of sensor nodes for $k = 2$, and 55% of sensor nodes for $k = 3$. Since the optimized network topologies for $k = 3$ contain more connections, it is expected that those networks will have a higher tolerance for node failures. On the other hand, more connections will consume more battery power, which will affect network lifetime. We therefore also examine lifetime measurements of the networks for the same set of scenarios.

Further, as we can observe in Figure 6.2, network topologies generated by the DPV algorithm become k-vertex supernode-disconnected after the failure of at most 1% of the sensor nodes. Even though the initial optimized topologies generated by DPV are k-vertex supernode-connected, after the failure of a few sensor nodes, the remaining topologies become at most $(k-1)$-vertex supernode-connected. The ADPV algorithm, on the other hand, maintains two-vertex supernode connectivity up to a failure of 32% of sensor nodes, and three-vertex supernode connectivity up to a failure of 21% of sensor nodes, on average among all experimental instances.

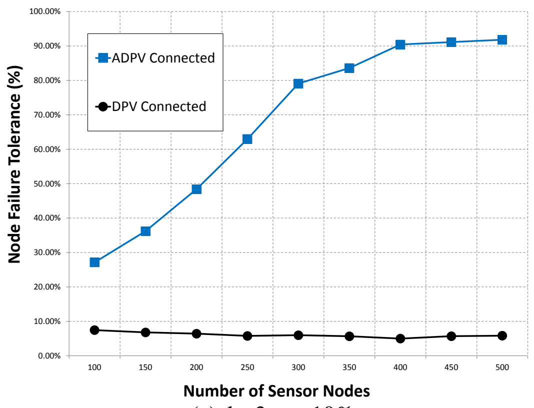
In Figures 6.3 and 6.4, we compare the lifetime measurements of the same set of experimental instances with those of Figure 6.1 and 6.2. A prominent aspect of ADPV is considering the sensor nodes' remaining energy levels; as a result, energy depletion occurs less frequently. This factor, when coupled with the adaptive nature of the



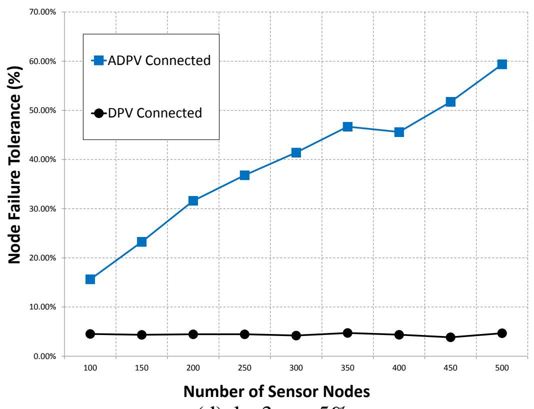
(a) k=2, sr=10%



(b) k=2, sr=5%

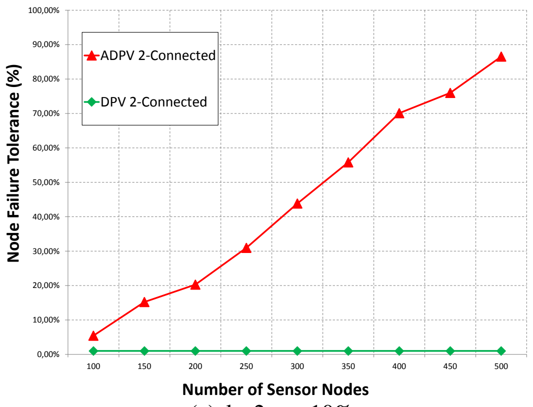


(c) k=3, sr=10%

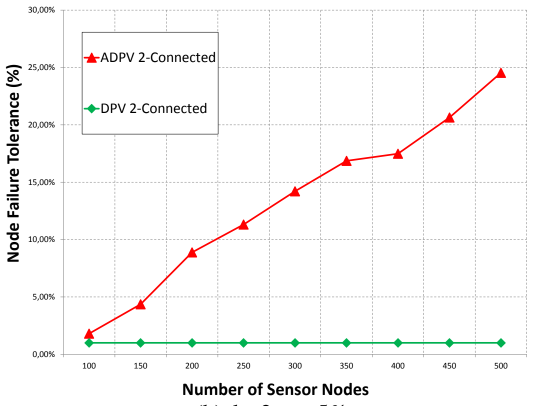


(d) k=3, sr=5%

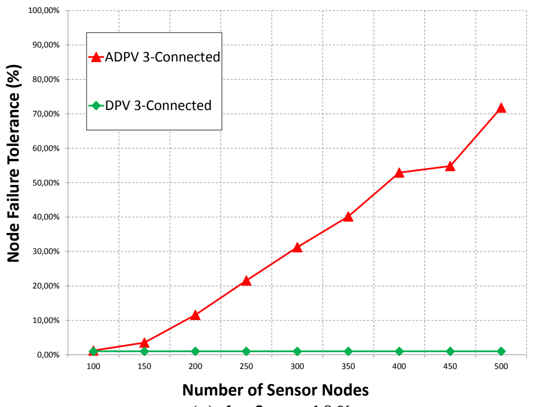
Figure 6.1: Percentage of failed sensor nodes when the network becomes supernode disconnected.



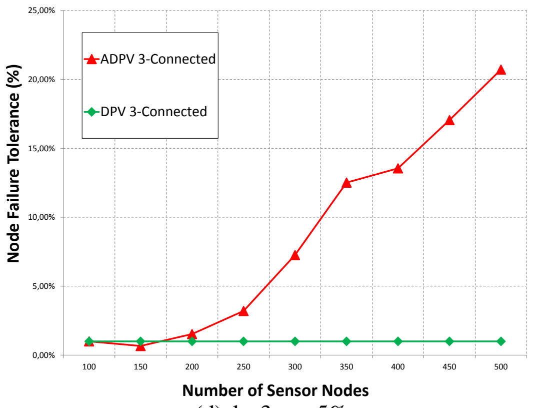
(a) k=2, sr=10%



(b) k=2, sr=5%



(c) k=3, sr=10%



(d) k=3, sr=5%

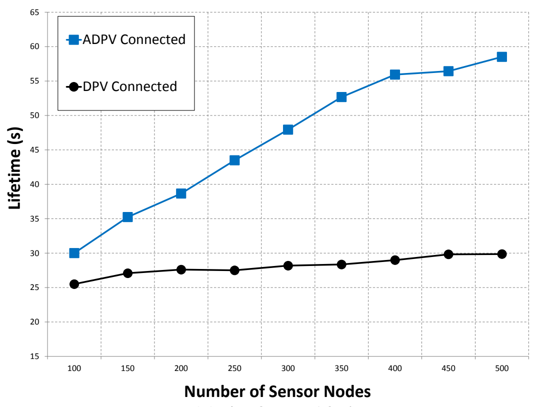
Figure 6.2: Percentage of failed sensor nodes when the network becomes *k*-vertex supernode disconnected.

algorithm, results in longer network lifetimes.

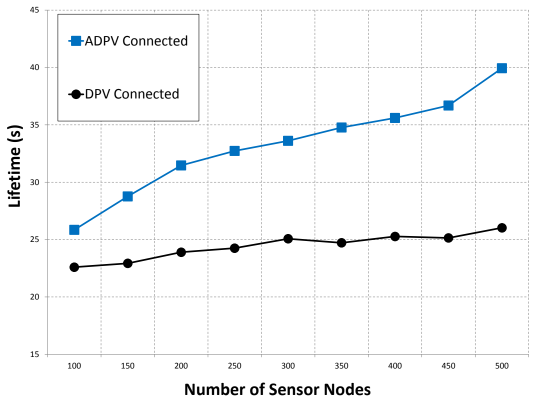
As we observe in Figure 6.3, in terms of one-vertex supernode-connected lifetimes, on average, ADPV results in a two-fold increase with respect to DPV. For the same experimental instances, as shown in Figure 6.4, ADPV provides respectively 65% and 46% longer two-vertex and three-vertex supernode-connected lifetimes than DPV. As we observe in the figures, network density has almost no effect on the lifetimes of the topologies generated by DPV. On the other hand, ADPV successfully prolongs the network lifetime almost proportionally to the network density for all $k = 1, 2, 3$.

We observe in Figures 6.4(c) and 6.4(d) that if the number of sensor nodes drops below a certain threshold (in our case, 150 sensor nodes in a 600m x 600m area when $sr = 10\%$, and 200 sensor nodes when $sr = 5\%$), it is hard to restore three-vertex supernode connectivity. This finding suggests that a minimum number of sensor nodes for every k and sr value is necessary to restore k-vertex supernode connectivity. Figures 6.2 and 6.3 respectively compare node failure tolerance and network lifetime for different values of $sr = 5\%, 10\%$ and $k = 2, 3$. According to the results, with the increasing number of supernodes, lifetime also increases, however, the relation between the increase in the number of supernodes and the increase in the lifetime is sublinear, and therefore, we expect that the increase in the lifetime becomes insubstantial as the number of supernodes exceeds a certain threshold. We also notice that, with the increasing k value, more disjoint paths are required and this makes providing alternative routes harder. In Figures 6.5(a) and 6.5(b), we compare DPV and ADPV algorithms for $k = 4$ in terms of network lifetime and node failure tolerance, respectively. As seen in Figure 6.5(a), ADPV successfully prolongs both one-vertex and four-vertex supernode-connected lifetimes of the network. Also, in Figure 6.5(b), we see that ADPV can preserve four-vertex supernode-connectivity up to the failure of 50% of the sensor nodes on dense networks, which in turn, achieves almost a two-fold increase in the four-vertex supernode-connected lifetime.

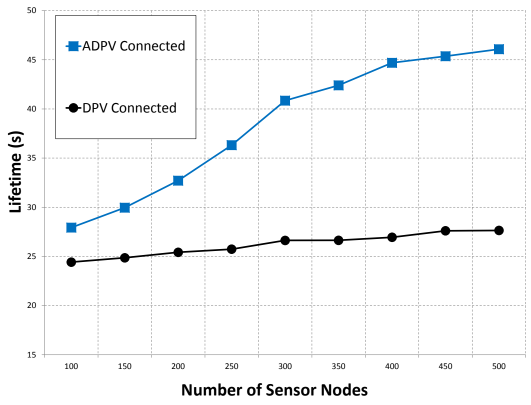
Another important metric we measure during our analysis is the number of message transmissions. Message transmission is an important metric because we must not only consider power consumption in the resulting topologies but also consider the power required to generate those topologies, which can be viewed as a fixed cost



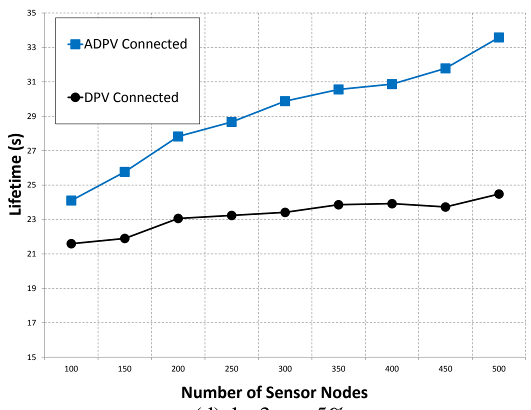
(a) k=2, sr=10%



(b) k=2, sr=5%

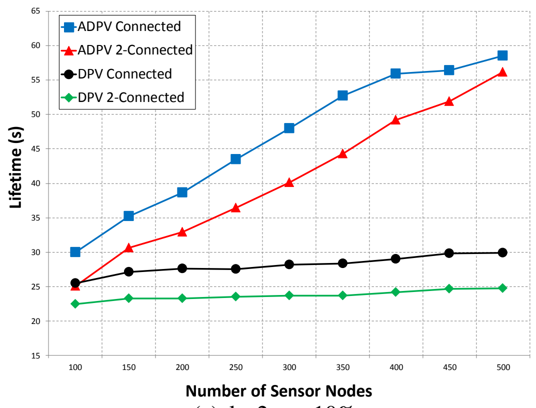


(c) k=3, sr=10%

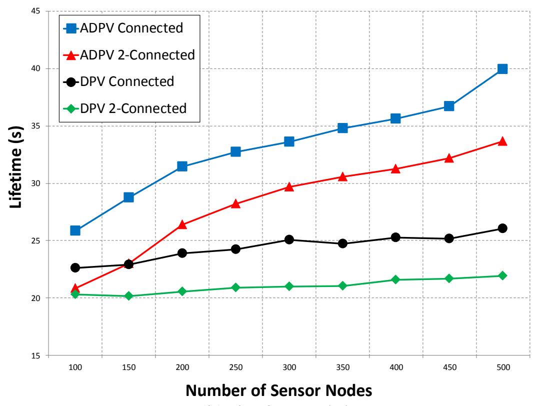


(d) k=3, sr=5%

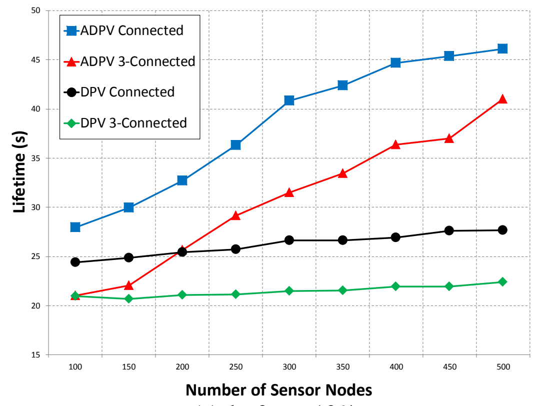
Figure 6.3: Connected lifetime comparison of the DPV and ADPV algorithms.



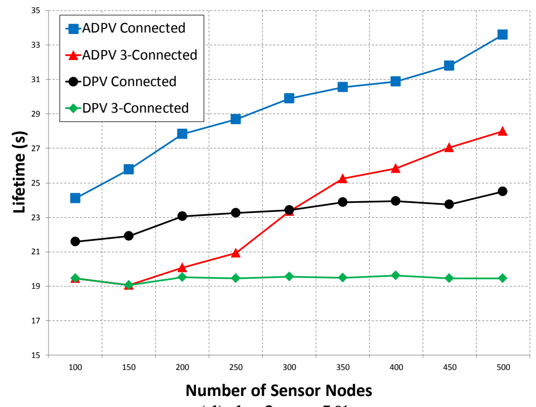
(a) k=2, sr=10%



(b) k=2, sr=5%

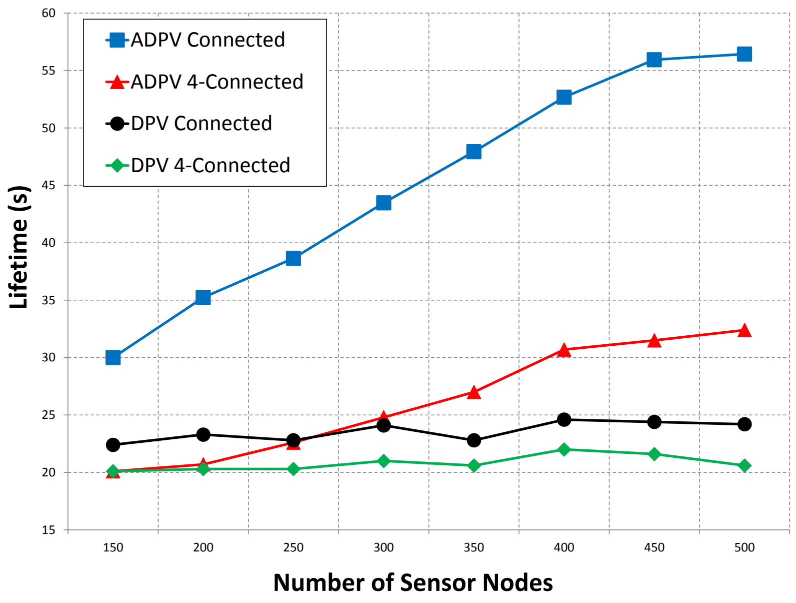


(c) k=3, sr=10%

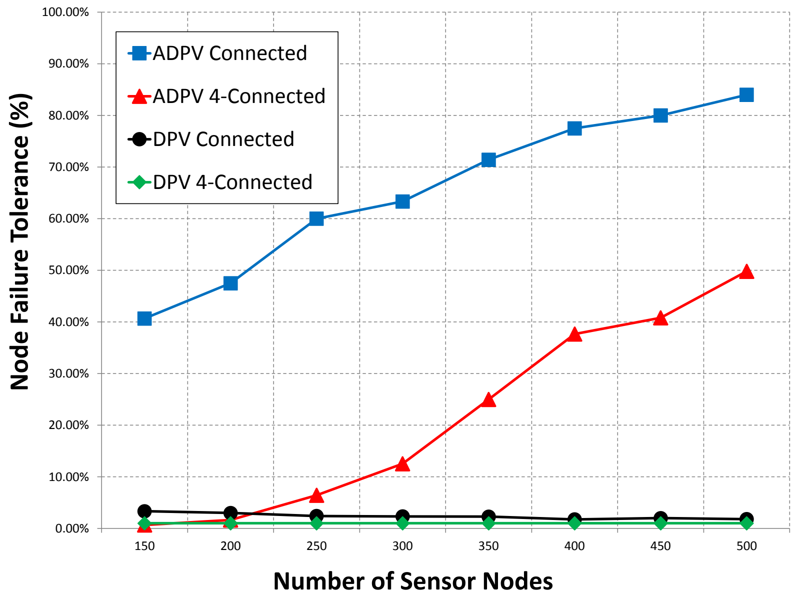


(d) k=3, sr=5%

Figure 6.4: 2- and 3-connected lifetime comparison of the DPV and ADPV algorithms.



(a) Lifetime comparison



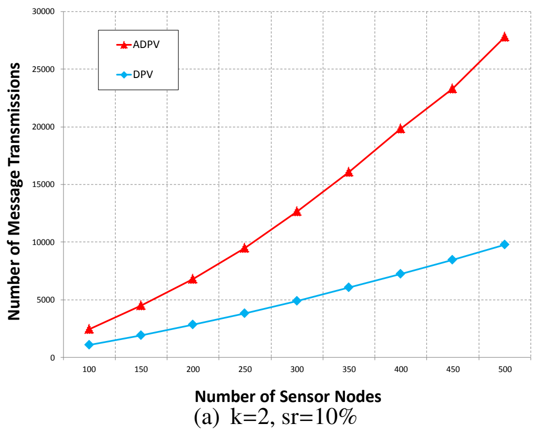
(b) Node failure tolerance comparison

Figure 6.5: Lifetime and node failure tolerance of DPV and ADPV algorithms for $k = 4$.

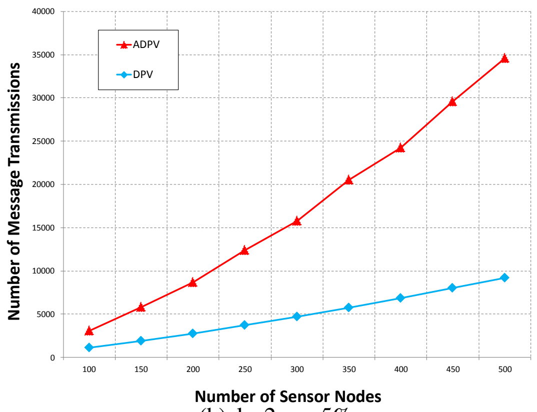
of obtaining the final topologies. If this cost is high, then the power efficiency of the resulting topology might become meaningless. In Figure 6.6, we compare the number of DPV and ADPV message transmissions and in Figure 6.7 we compare ratio of these message counts. To simulate the worst-case scenario of ADPV, we set the waiting period in the restoration phase to zero, which means that every node failure that affects disjoint paths will trigger a restoration phase for that node. According to the results, for $k = 2$, ADPV makes at least 2.25 times and at most three times the message transmissions than DPV does, and for $k = 3$, ADPV makes at least three times and at most 3.5 times the message transmissions of DPV. As seen in the sub-figures of Figure 6.6, the number of message transmissions of both algorithms increases almost linearly with the number of sensor nodes, but as seen in Figure 6.7 the ratio of these message counts does not significantly change. This result is also compatible with the expected number of message transmissions. As it is discussed in Section 4.6, in terms of number of transmitted messages ADPV and DPV is expected to have the same message complexity and with the simulation results this expectation is also verified.

In Figure 6.8 and Figure 6.9, we can see total number of connectivity restorations in ADPV algorithm for different k values and supernode densities. Our first observation according to these results is, number of connectivity restorations is proportional with the sensor node density. With the increasing number of sensor nodes, number of connectivity restorations also increase. Our second observation is, number of connectivity restorations is higher for lower supernode densities. The reason for this result is, with less supernodes, there exist more critical sensor nodes which break supernode connectivity with their battery depletion and therefore more connectivity restoration phase takes place. Another and probably the most important observation is, as seen in Figure 6.9, when $k = 3$ ADPV restores supernode connectivity almost 350 times and more importantly as seen in Figure 6.7 ADPV achieves it with only 3.5 times the message transmissions compared to DPV algorithm. Considering the extra messages in ADPV are used for updating the residual energy levels of the remaining active sensor nodes in the disjoint paths, these messages are very crucial for better load balancing and for making maximum use of the available sensor nodes.

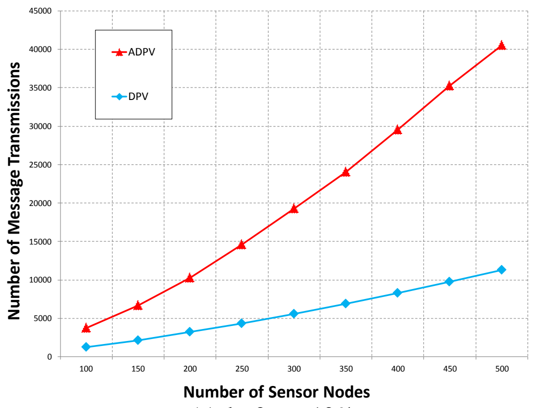
In Figure 6.9, we observe that number of connectivity restorations for $k = 3$ is always



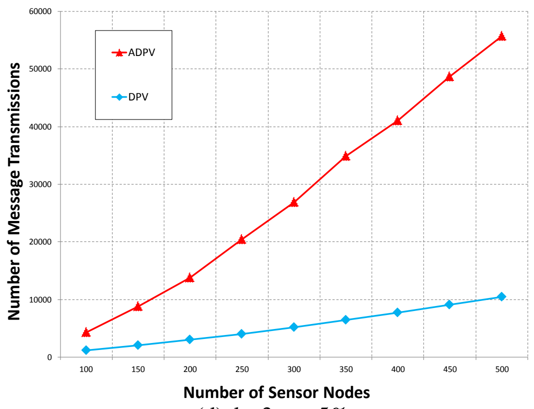
(a) k=2, sr=10%



(b) k=2, sr=5%



(c) k=3, sr=10%



(d) k=3, sr=5%

Figure 6.6: Number of message transmissions in DPV and ADPV algorithms.

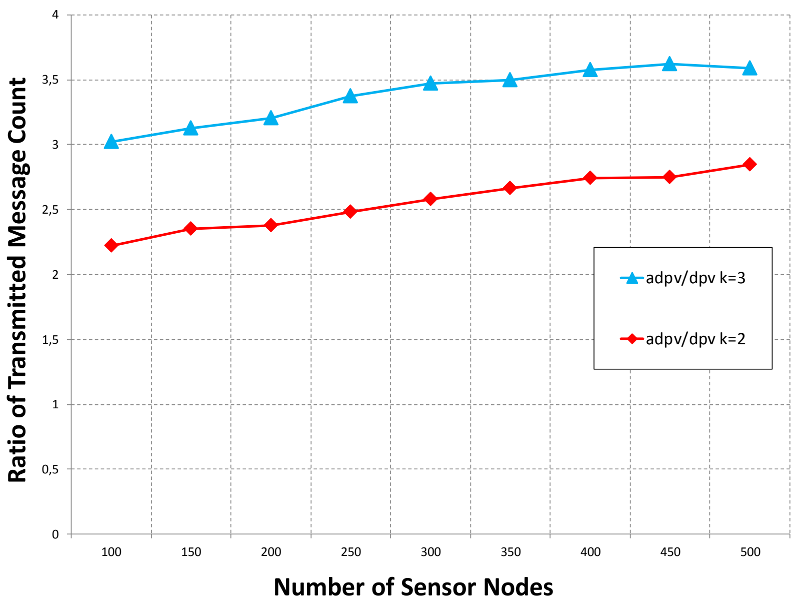
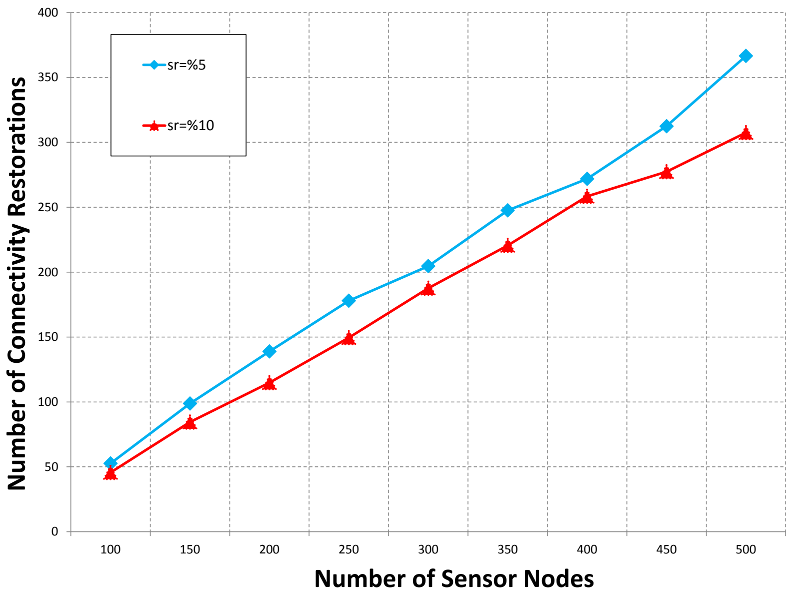


Figure 6.7: Ratio of transmitted messages in DPV and ADPV algorithms.

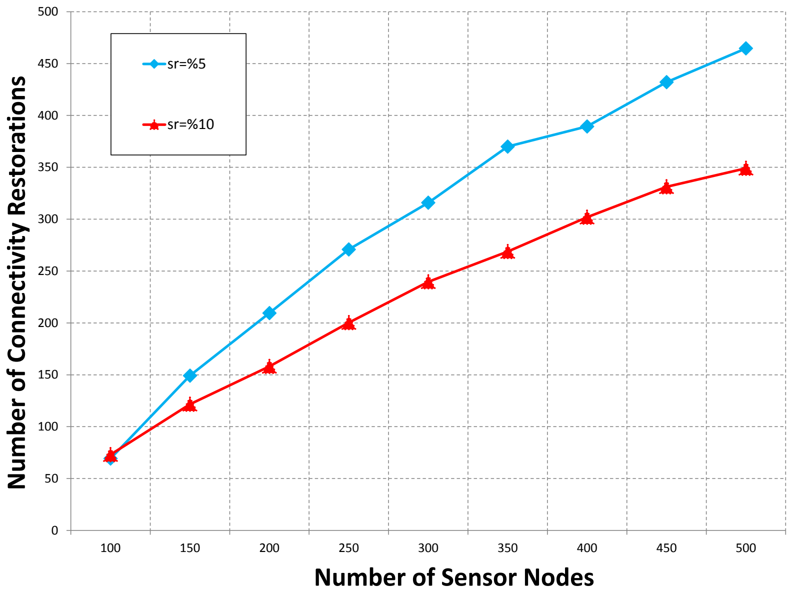
higher than for $k = 2$, but the difference in number of connectivity restorations does not significantly change when network density changes and the resulting graphs are parallel to each other. We also observe that, for both k values, number of connectivity restorations are proportional with the sensor node count. As we can see more clearly in Figure 6.10, number of connectivity restorations per node does not significantly change with the increasing sensor count. In the worst case scenario, which occurs when $k = 3$ and $sr = \%5$, number of connectivity restorations per node in the average equals to 1.00. This means, each node executed connectivity restoration process only once in the average and as a result, a two-fold increase in the supernode-connected lifetime of the network is obtained.

In Figure 6.10, we can also observe that, average number of connectivity restorations increases with the increasing k values and decreasing supernode ratios. Since ADPV restores connectivity whenever connectivity degree gets below k, it is an expected result to have more connectivity restorations for higher k values. For the decreasing supernode counts, there will be more critical nodes, that will break connectivity with its battery depletion, therefore will result more connectivity restoration processes.

In order to observe the optimality of ADPV, we also compared it with multiple DPV



(a) k=2



(b) k=3

Figure 6.8: Number of successful connectivity restorations for different supernode counts.

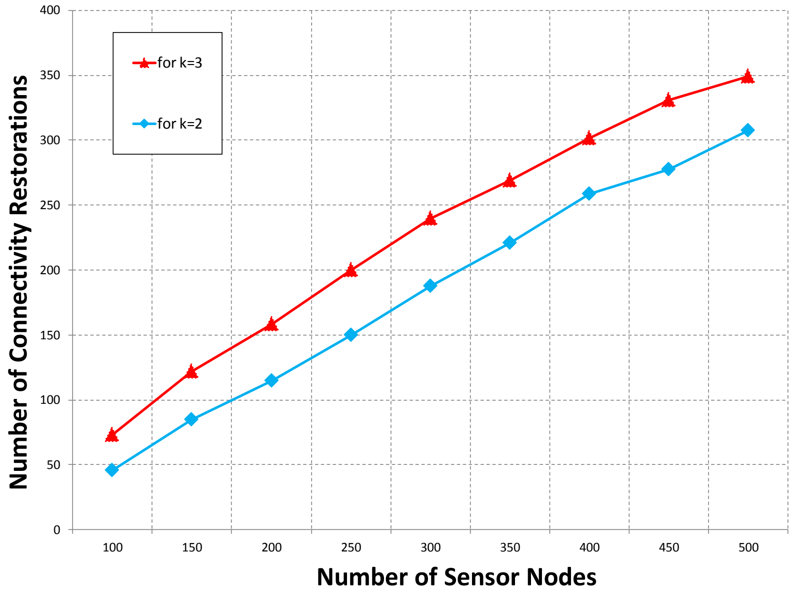


Figure 6.9: Number of successful connectivity restorations for different *k* values.

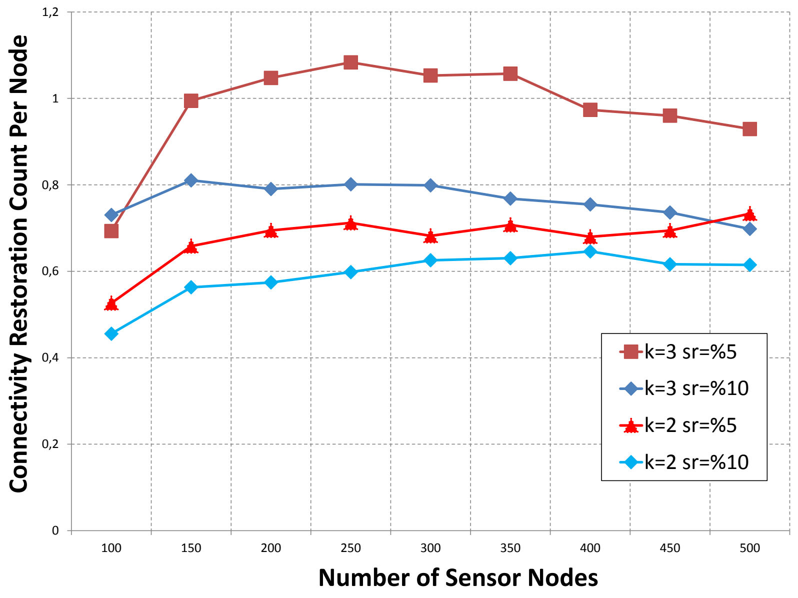


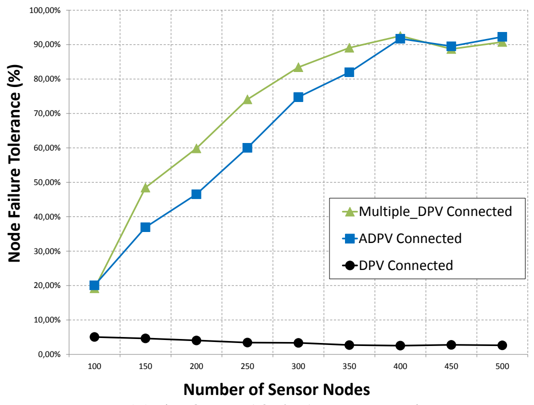
Figure 6.10: Connectivity restoration count per node.

executions. In the multiple DPV case, we assume DPV algorithm has no cost and whenever networks supernode connectivity is broken, we set transmission ranges of all the remaining sensor nodes to R_{max} and execute DPV from scratch. Since DPV makes sure it will calculate a k-vertex supernode-connected sub-network from the given k-vertex supernode-connected network, as a result of executing DPV algorithm a new network topology that contains the remaining sensor nodes and if possible is k-vertex supernode-connected is calculated. Although, compared to ADPV, DPV has very high cost and make lots of message transmissions and changes the global network topology, we ignore all these costs and observe up to what point DPV can preserve 1- and 2-vertex supernode connectivity. For instance, if every sensor node failure breaks supernode-connectivity in a 500 sensor node network, after each sensor node failure DPV will be executed in total of 499 times.

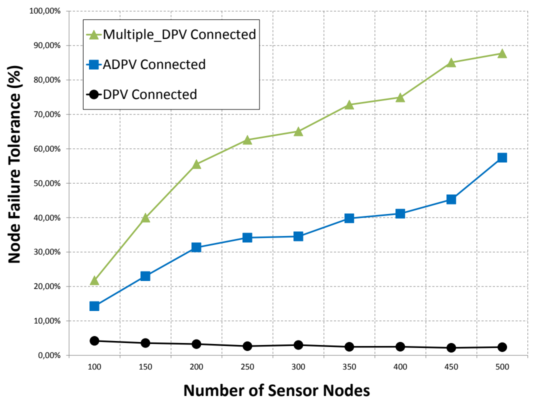
In Figure 6.11, node failure tolerance of ADPV and multiple DPV executions is compared. As expected, multiple DPV preserves network's supernode-connectivity longer than ADPV does. However, when supernode-ratio equals to 10%, as shown in Figures 6.11(a) and 6.11(c) this difference is less than 10% and this shows the robustness of ADPV algorithm. Although ADPV makes local changes, it can preserve the global supernode-connectivity for sufficiently long time, that is almost comparable with the multiple executions of DPV algorithm. In Figures 6.11(b) and 6.11(d), we can observe the results for less supernodes. According to these results, we can observe the success of multiple DPV does not significantly change with the changing supernode counts, but ADPV's does. With the decreasing supernode count, the difference between ADPV and multiple DPV increases up to 50%.

In addition to fault-tolerant lifetimes, in order to show ADPV maintains the aim of optimizing total transmission power we also evaluated average and maximum lifetimes of the given networks. ADPV adapts network structure to node failures and achieves this by choosing paths which are not initially preferred. Therefore, one may expect ADPV to have worse results than DPV for these criteria. However, as it can be derived from Figure 6.12 and Figure 6.13 while choosing initially not preferred paths, ADPV still considers the aim of optimizing total transmission power.

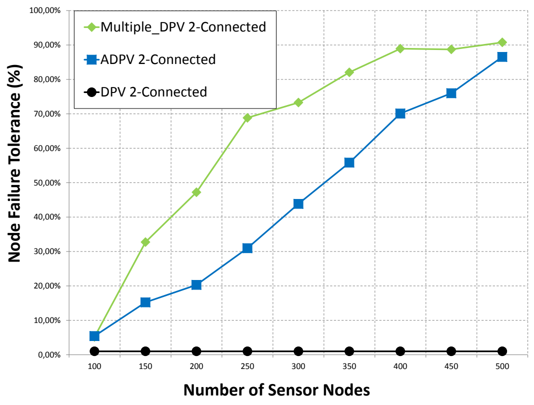
In terms of average lifetimes, which equals to the arithmetic mean of the lifetime of



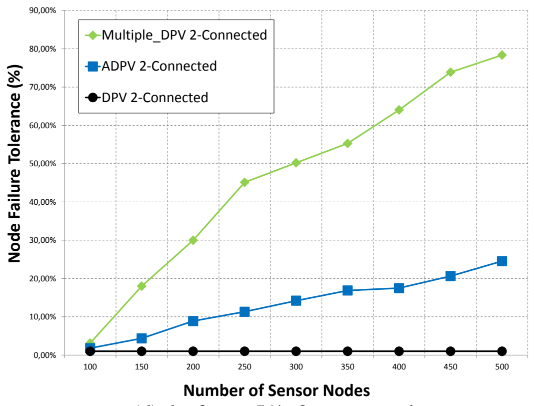
(a) k=2, sr=10%, 1-connected



(b) k=2, sr=5%, 1-connected



(c) k=2, sr=10%, 2-connected



(d) k=2, sr=5%, 2-connected

Figure 6.11: Comparison with multiple DPV execution.

sensor nodes, DPV and ADPV algorithms seems to have similar results, which are both better than UDG algorithm. Since, for UDG algorithm, when there are more nodes in range to communicate with, there will obviously be more power consumption, this is an expected result. DPV and ADPV removes non-required connections and as a result an increase in network density does not have a big impact on the average lifetime performance of these approaches.

In terms of maximum lifetimes, which shows the elapsed time until the last node dies, DPV and ADPV algorithms have similar results and which is better than Unit Disk Graph algorithm. Also, ADPV first changes network topology after first node dies. Therefore, in terms of lifetime of first dying node, DPV and ADPV exact same results.

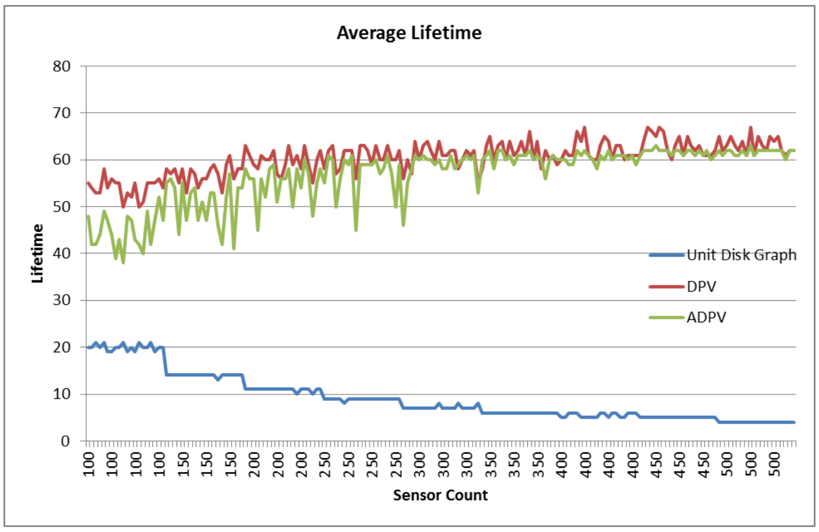


Figure 6.12: Average lifetimes.

In order to observe the success of ADPV in terms of load balancing, we compared it with Low Energy Adaptive Clustering Hierarchy (LEACH) algorithm [34]. LEACH is a clustering based protocol that utilizes randomized rotation of local cluster heads and the main purpose of the protocol is even distribution of energy load among all sensor nodes. The operation of LEACH is broken up into rounds and in each round nodes decide whether or not to become a cluster-head for the current round. This decision is based-on the suggested percentage (p) of cluster-heads for the network and the last time the node was a cluster-head. The most common cluster-head percentages

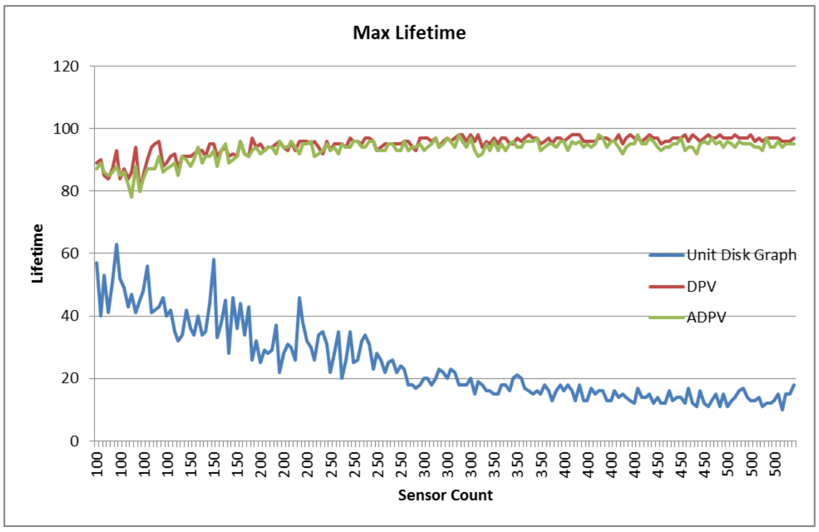


Figure 6.13: Maximum Lifetimes

are $p = 0.05$ and $p = 0.10$. As it can be seen in Figure 6.14, in terms of connected-lifetimes ADPV performs better than LEACH algorithm. Considering the success of LEACH algorithm in balancing the energy load, we can conclude ADPV is also quite successful in load balancing the energy consumption among sensor nodes.

6.2 Experimental Results for MSDPV Algorithm

In this section we report our measurements regarding the minimum number of supernodes required to maintain k-vertex supernode connectivity. We also discuss the contribution of placing supernodes at known locations to the network lifetime compared to random distribution of supernodes. We implemented MSDPV using an extended version of a custom simulator, which has also been used for evaluating DPV and ADPV algorithms. This framework has the ability to measure supernode-connected and k-vertex supernode-connected lifetimes of the given heterogeneous WSNs. In order to measure the lifetime with random and uniform distribution of supernodes, we use the ADPV algorithm with random supernode locations and represent it as ADPV with random supernode locations (ADPVR) throughout this section.

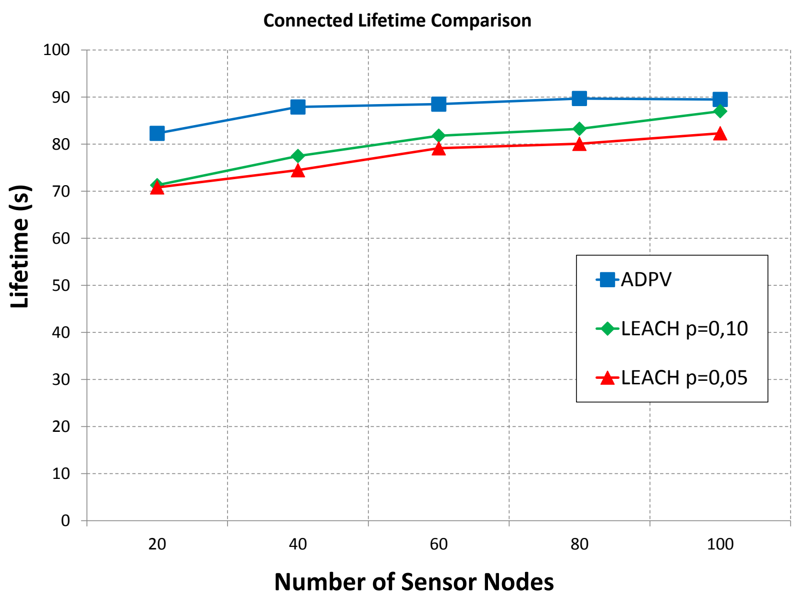


Figure 6.14: Lifetime comparison of ADPV and LEACH algorithms.

6.2.1 Experimental Setup

In our experiments, we vary the number of sensor nodes in the network between 100 to 500, and assume they are uniformly and randomly deployed in a 600m x 600m area. We assumed maximum communication range of sensor nodes to be equal to 100m, and for the degree of disjoint connectivity we executed the simulations for both $k = 2$ and $k = 3$. For each message transmission we assumed a packet loss rate of 10% and pathloss exponent of 2. Finally, we repeated our experiments for 20 times and report the averages. Our simulation parameters are summarized in Table 6.2. For each set of scenario, we first determine the minimum number of supernodes and their locations using the MSDPV algorithm. Then by randomly relocating the supernodes we generate another network topology and measure lifetimes of both topologies.

6.2.2 Results

In Figure 6.15, we present the results of MSDPV algorithm and also the main aim of this study regarding the minimum number of supernodes required to maintain two- and three-vertex supernode connected network topology. According to the results,

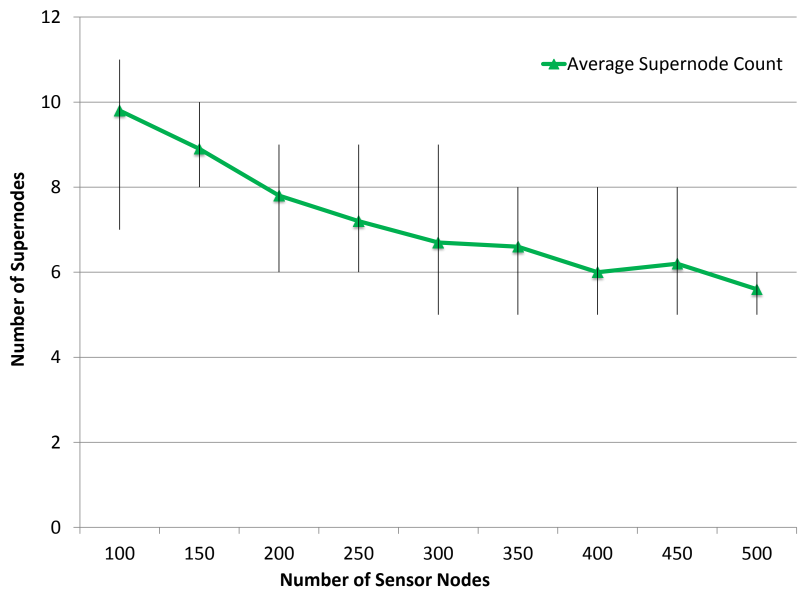
Table 6.2: Simulation parameters for MSDPV.

Number of Sensor Nodes: N	$[100 \ldots 500]$
Deployment Area	600m x 600m
Initial Transmission Range of Sensor Nodes: R_{max}	100m
Degree of Disjoint Connectivity: k	2 and 3
Packet Loss Rate	10%
Power Attenuation Exponent	2

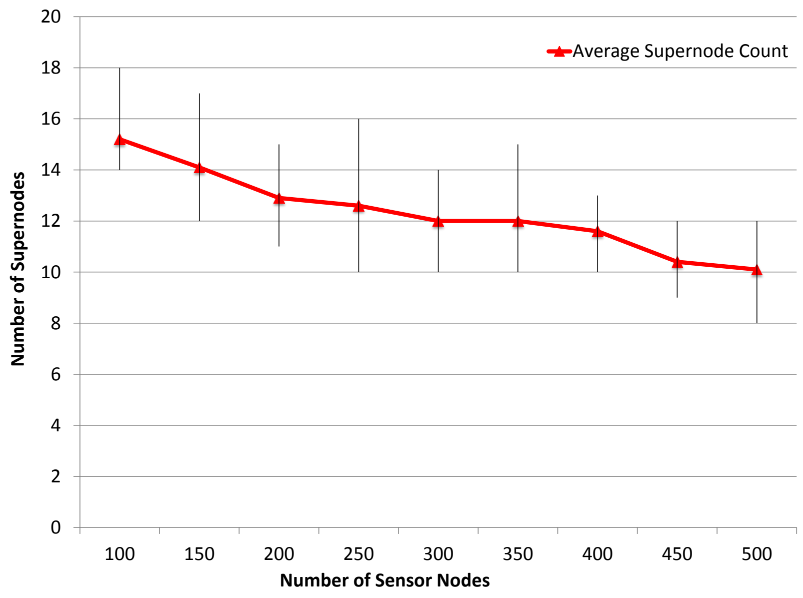
our first observation is that with the increasing k values, number of required supernodes also increases. Since larger k values require more disjoint paths from the sensor nodes to the set of supernodes, this is an expected behavior. Our second observation is that number of sensor nodes is inversely proportional to the required number of supernodes. With the increasing number of sensor nodes, required supernode counts to maintain k-vertex supernode connectivity decreases for both $k = 2$ and $k = 3$. This is a quite interesting result. When we examine the experimental instances in [10], [6] and [18] with the increasing number of sensor nodes, they also increase number of supernodes. However, the results in Figure 6.15 indicate that with the increasing number of supernodes it gets easier to find alternative routes and less number of supernodes is sufficient to provide k-vertex supernode connectivity. Because of higher number of alternative routes this is also an expected behavior and the results of MSDPV copes with the expected behavior.

For dense networks, we can observe that number of required supernodes to provide higher network connectivity is quite small. For instance, in one extreme, when we examine the results of a 500-node network, for $k = 2$, MSDPV determines number of required supernodes to be about 5. Even though this number may not be the minimum, it shows MSDPV is quite successful in determining almost minimum number of supernodes and their locations. In other extreme for sparser networks with higher k values, when we examine 100-node network for $k = 3$, MSDPV determines number of required supernodes to be about 15.

In Figure 6.16 and 6.17, we compare the supernode-connected and k-vertex supernode connected lifetimes of the topologies generated by MSDPV and ADPVR algorithms and observe the most striking result. Note that, for this evaluation we consider



(a) $k = 2$



(b) $k = 3$

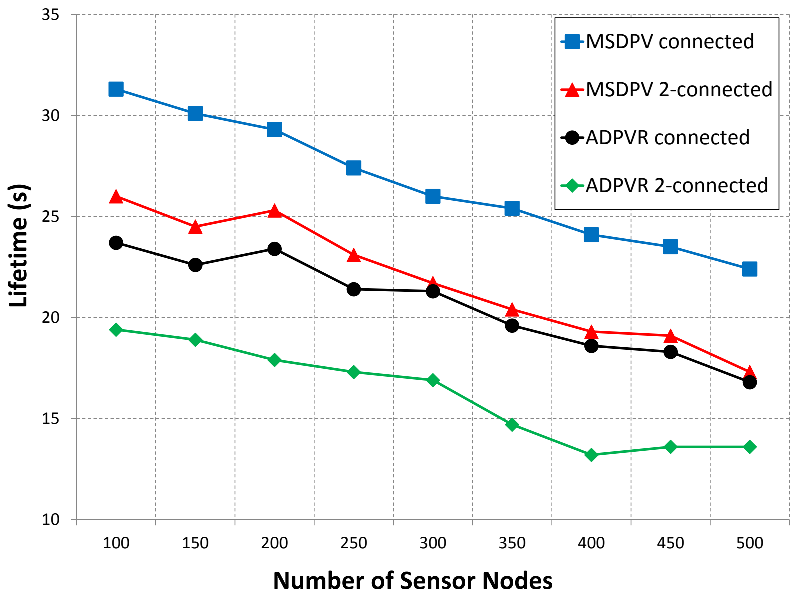
Figure 6.15: Number of required supernodes.

the first time supernode-connectivity and k-vertex supernode connectivity is broken. We observe that, just by changing the locations of the supernodes, MSDPV can improve both supernode-connected and k-vertex supernode connected lifetimes of the networks. For $k = 2$, MSDPV shows 29% improvement and for $k = 3$, MSDPV shows 22% improvement, in the average, in supernode-connected lifetimes (Figure 6.16) of the network compared to random deployment of supernodes.

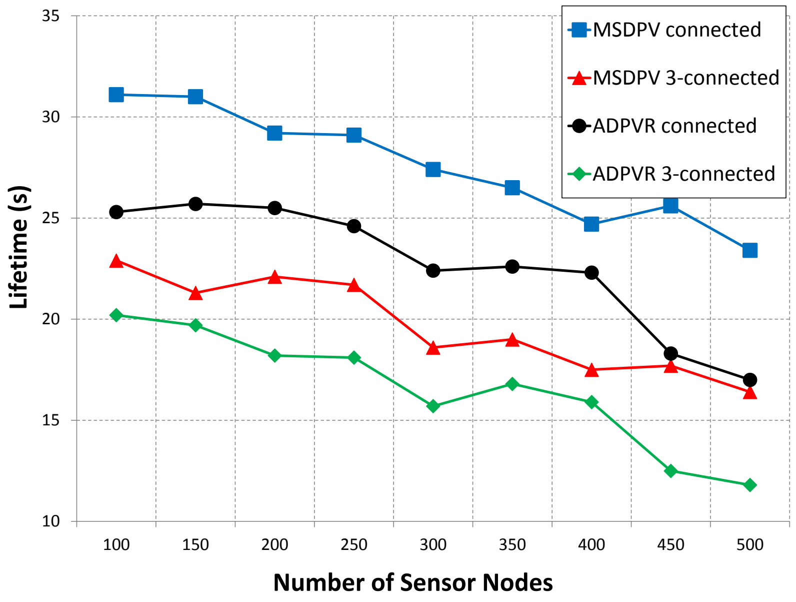
Also, as shown in Figure 6.17, for the same experimental scenarios, MSDPV provides respectively 35% and 19% longer two-vertex and three-vertex supernode-connected lifetimes, in the average, than random deployment of supernodes. Therefore, we can infer that, MSDPV does not only calculate minimum number of required supernodes, but also proposes a good method for the places of supernodes to increase connected lifetime of the network compared to uniform distribution.

Since sensor nodes consume most of their battery power for the transmission of messages, another important metric we need to consider is the number of required message transmissions. Figure 6.18 shows the total number of required message transmissions to execute the MSDPV and ADPVR algorithms. Since ADPVR algorithm randomly determines supernode locations without making any message transmissions, number of message transmissions of ADPVR algorithm, as expected, is lower than MSDPV algorithm. However, for all experimental instances, this increase is linear. As shown in Figures 6.18(a) and 6.18(b), for $k = 2$ and $k = 3$, MSDPV incurs a 100% and 50% increase on the required number of message transmissions compared to ADPVR, respectively. It does not increase the complexity of message count, but in the worst case, doubles the number of transmitted messages.

Another important observation is that, as expected, number of message transmissions in MSDPV algorithm does not significantly change with the increasing k value. Since MSDPV makes message transmissions during the first phase, in which k value is not being used as an input, number of message transmissions does not get effected by the increasing k value. In this phase, all sensor nodes are assumed to be candidate supernodes and find alternative paths to other candidate supernodes. As it is also expected theoretically, experimental results shown in Figure 6.19 suggest that with the increasing k value number of message transmissions of MSDPV algorithm does

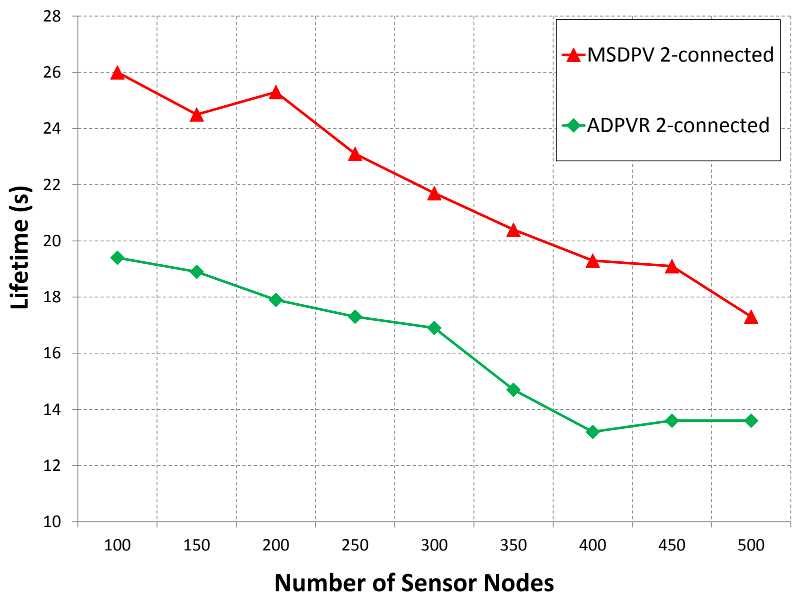


(a) $k = 2$

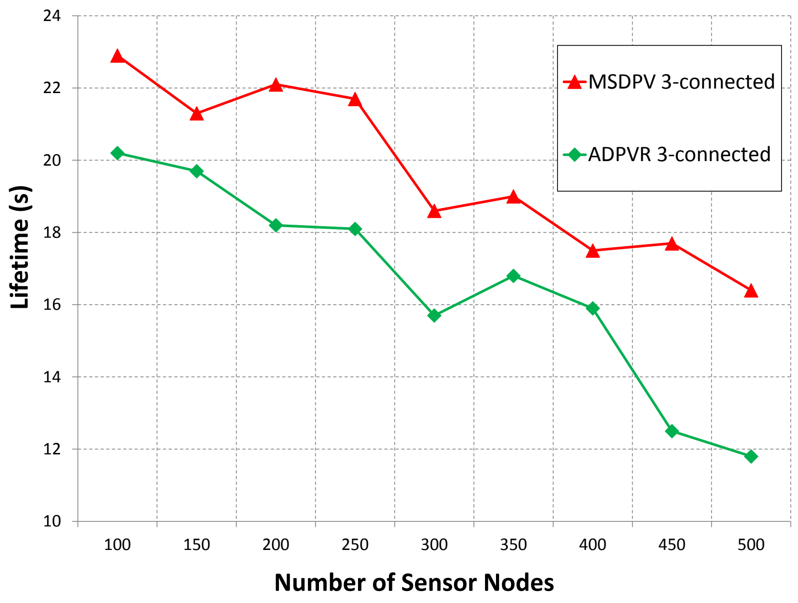


(b) $k = 3$

Figure 6.16: Lifetime comparison of the MSDPV and ADPVR algorithms.

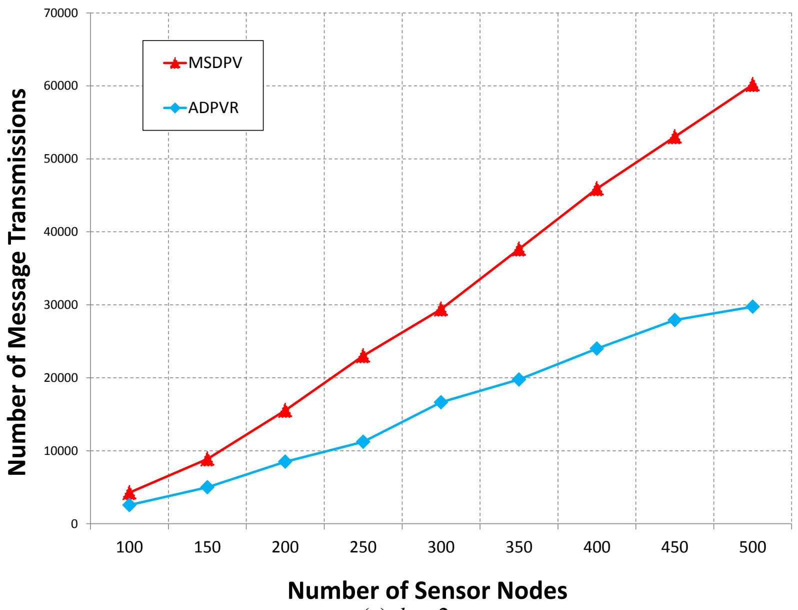


(a) $k = 2$

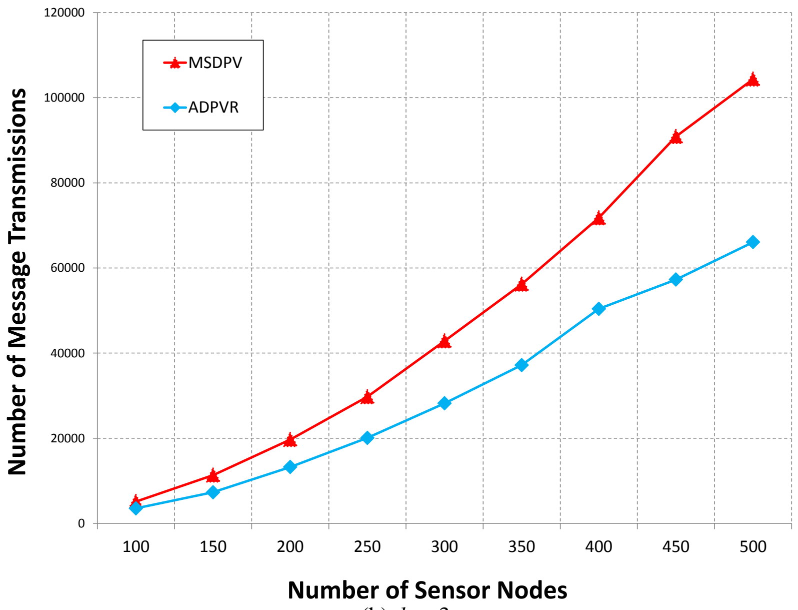


(b) $k = 3$

Figure 6.17: Lifetime comparison of the MSDPV and ADPVR algorithms.



(a) $k = 2$



(b) $k = 3$

Figure 6.18: Number of message transmissions in MSDPV and ADPVR algorithms.

not differ. Therefore, with the increasing *k* value, MSDPV brings less overhead over ADPVR.

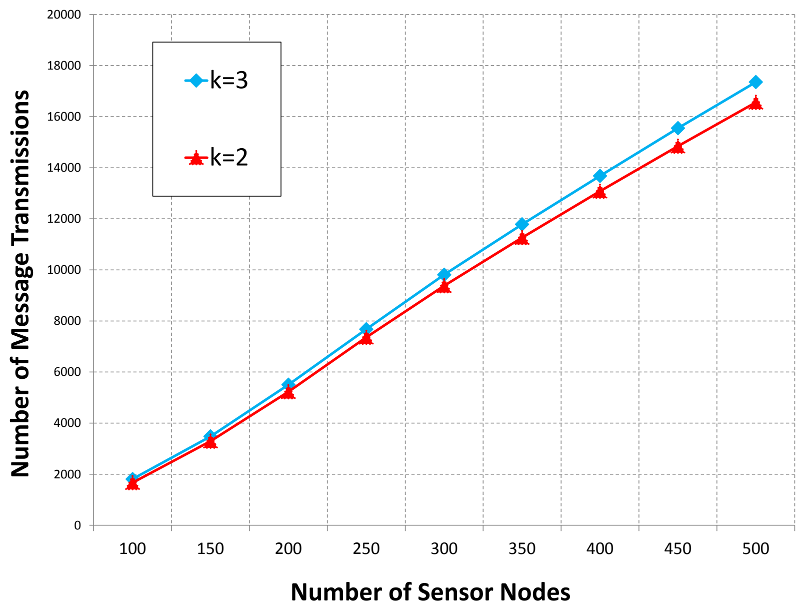


Figure 6.19: Number of message transmissions during the first phase of MSDPV algorithm.

CHAPTER 7

CONCLUSION

Wireless sensor networks (WSNs) have been widely recognized as a promising technology and studied extensively for their advantages over traditional communication technologies including their low cost and easy deployment without any infrastructure. WSNs have their unique challenges and requirements such as energy efficiency and fault-tolerant operation. In many applications, WSNs operate in inhospitable and harsh environments, such as battlefields and forests, where the nodes are subject to increased risk of getting damaged. Also, nodes are equipped with small batteries and their operation ceases upon depleting of their energy supply. Therefore, solutions to prolong network lifetime and provide fault-tolerance is of considerable importance.

Recent studies have revealed that heterogeneous WSNs, where the network consists of different kind of sensors can result better energy efficiency and a higher fault-tolerance compared to homogeneous wireless sensor networks [97]. In this work, we focus on heterogeneous two-layered network architectures where the lower layer consists of ordinary sensor nodes and the upper layer consists of resource-rich supernodes. In such topologies, sensor nodes forward their data towards supernodes using multi-hop paths. Supernodes collect and process the incoming data and can make critical decisions based on the application. For some scenarios, data delivery can be critical, so a fault-tolerant topology would be essential where each node has a certain degree of connectivity to the set of supernodes.

In this study, we present two algorithms, namely ADPV and MSDPV. ADPV is an adaptive, energy-aware and distributed topology-control algorithm. The motivation of this algorithm is to prolong the supernode-connected lifetime of given heterogeneous

WSNs. The ADPV algorithm consists of two phases: initialization and restoration. During initialization, ADPV computes alternative routes in the network. To determine routes efficiently ADPV employs a novel method based on set packing. Whenever *k*-vertex supernode connectivity is broken, the restoration phase is activated. To restore connectivity, ADPV utilizes those alternative routes and adjusts the sensor nodes' transmission ranges accordingly. The ADPV algorithm is a distributed algorithm in both the initialization and restoration phases. A broad set of conducted simulations agrees well with the theoretical anticipation that ADPV can significantly prolong supernode-connected lifetimes of heterogeneous WSNs. Our adaptive algorithm increases the durability of network connectivity against node failures, from 5% up to 95%. As for *k*-vertex connectivity, we are able to keep the network two- and three-vertex supernode-connected up to the failure of 90% and 75% of sensor nodes, respectively.

In ADPV, we assume that supernodes are stationary and arbitrarily deployed with no concern for sensor node positions. MSDPV algorithm relaxes these assumptions and when together used, they form a more concrete framework to be used in heterogeneous WSNs with fault-tolerance requirements. MSDPV a distributed and energy-aware approach to locate minimum number of supernodes to maintain *k*-vertex supernode connectivity. The motivation of this algorithm is to optimize supernode usage in heterogeneous WSNs and also to prolong supernode-connected network lifetime. MSDPV places supernodes with respect to the positions of already-deployed sensor nodes using an optimization that is based on the well-known set-cover problem. In this way, MSDPV uses the opportunity to find more center-like positions within sensor nodes, and in turn improve system efficiency. Experimental results for the MSDPV algorithm validates our theoretical expectations and the number of supernodes required to make topology *k*-vertex supernode connected decreases as the sensor node density increases. With the extensive simulations, we demonstrate that MSDPV can significantly prolong supernode-connected lifetime of the network up to 40% compared to uniform distribution of supernodes.

In this study, we assume that supernodes are stationary and deployed during the initialization phase of the network. It would be an interesting future work to relax these assumptions and instead of being stationary, supernodes can be mobile and thus could be repositioned to further increase network lifetime. In the literature, there is no work to restore *k*-connectivity using mobile nodes in a distributed and efficient manner [101]. Also, in this study, fault-tolerance is achieved for ordinary sensor nodes and it would also be an interesting future work to provide some level of fault-tolerance for supernodes together with the ordinary sensor nodes.

REFERENCES

[1] A. A. Abbasi, K. Akkaya, and M. Younis. A distributed connectivity restoration algorithm in wireless sensor and actor networks. In *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pages 496–503. IEEE, 2007.

[2] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag. Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility. *Computers, IEEE Transactions on*, 59(2):258–271, Feb 2010.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Comput. Netw.*, 38(4):393–422, Mar. 2002.

[4] G. Anastasi, M. Conti, M. Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537 – 568, 2009.

[5] M. Azharuddin and P. K. Jana. A ga-based approach for fault tolerant relay node placement in wireless sensor networks. In *Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on*, pages 1–6, Feb 2015.

[6] H. Bagci, I. Korpeoglu, and A. Yazici. A distributed fault-tolerant topology control algorithm for heterogeneous wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1–1, 2014.

[7] A. Bari, A. Jaekel, J. Jiang, and Y. Xu. Design of fault tolerant wireless sensor networks satisfying survivability and lifetime requirements. *Computer Communications*, 35(3):320 – 333, 2012.

[8] A. Bogdanov, E. Maneva, and S. Riesenfeld. Power-aware base station positioning for sensor networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[9] M. Cardei and D.-Z. Du. Improving wireless sensor network lifetime through power aware organization. *Wireless Networks*, 11(3):333–340, 2005.

[10] M. Cardei, S. Yang, and J. Wu. Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 19(4):545–558, April 2008.

[11] A. Cerpa and D. Estrin. Ascent: adaptive self-configuring sensor networks topologies. *Mobile Computing, IEEE Transactions on*, 3(3):272–285, July 2004.

[12] Y. Chen and Q. Zhao. On the lifetime of wireless sensor networks. *Communications Letters, IEEE*, 9(11):976–978, Nov 2005.

[13] X. Cheng, D. Du, L. Wang, and B. Xu. Relay Sensor Placement in Wireless Sensor Networks. *Wireless Networks*, 14(3):347–355, 2008.

[14] S. Coleri, S. Cheung, and P. Varaiya. Sensor networks for monitoring traffic. In *In Allerton Conference on Communication, Control and Computing*, 2004.

[15] D. R. Dandekar and P. R. Deshmukh. Energy balancing multiple sink optimal deployment in multi-hop wireless sensor networks. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 408–412. IEEE, 2013.

[16] D. Das, Z. Rehena, S. Roy, and N. Mukherjee. Multiple-sink placement strategies in wireless sensor networks. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–7. IEEE, 2013.

[17] D. Das, Z. Rehena, S. Roy, and N. Mukherjee. Multiple-sink placement strategies in wireless sensor networks. In *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–7. IEEE, 2013.

[18] F. Deniz, H. Bagci, I. Korpeoglu, and A. Yazıcı. An adaptive, energy-aware and distributed fault-tolerant topology-control algorithm for heterogeneous wireless sensor networks. *Ad Hoc Networks*, 44:104 – 117, July 2016.

[19] S. S. Dhillon and K. Chakrabarty. *Sensor placement for effective coverage and surveillance in distributed sensor networks*, volume 3. IEEE, 2003.

[20] R. Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[21] E. J. Duarte-Melo and M. Liu. Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 1, pages 21–25. IEEE, 2002.

[22] M. Durisic, Z. Tafa, G. Dimic, and V. Milutinovic. A survey of military applications of wireless sensor networks. In *Embedded Computing (MECO), 2012 Mediterranean Conference on*, pages 196 –199, june 2012.

[23] S. Even. An algorithm for determining whether the connectivity of a graph is at least k. *SIAM Journal on Computing*, 4(3):393–396, 1975.

[24] G. Fan and S. Jin. Coverage problem in wireless sensor network: A survey. *Journal of networks*, 5(9):1033–1040, 2010.

[25] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[26] D. Ghataoura, J. Mitchell, and G. Matich. Networking and application interface technology for wireless sensor network surveillance and monitoring. *Communications Magazine, IEEE*, 49(10):90–97, Oct 2011.

[27] K. Haeyong, K. Taekyoung, and M. Pyeongsoo. Multiple sink positioning and routing to maximize the lifetime of sensor networks. *IEICE transactions on communications*, 91(11):3499–3506, 2008.

[28] S. Halder, A. Ghosal, and S. Bit. A pre-determined node deployment strategy to prolong network lifetime in wireless sensor network. *Computer Communications*, 34(11):1294 – 1306, 2011.

[29] M. Hammoudeh and R. Newman. Adaptive routing in wireless sensor networks: Qos optimisation for enhanced application performance. *Information Fusion*, 22:3–15, 2015.

[30] X. Han, X. Cao, E. Lloyd, and C.-C. Shen. Fault-Tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656, 2010.

[31] E. Hazan, S. Safra, and O. Schwartz. On the complexity of approximating k-set packing. *Comput. Complex.*, 15(1):20–39, May 2006.

[32] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1 –6, oct. 2007.

[33] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, Jan 2000.

[34] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *International Conference on System Sciences*, January 2000.

[35] W. Heinzelman, A. Chandrakasan, and A. Smith. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1:660–670, 2002.

[36] Y. K. Joshi and M. Younis. Autonomous recovery from multi-node failure in wireless sensor network. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 652–657. IEEE, 2012.

[37] Y. K. Joshi and M. Younis. Restoring connectivity in a resource constrained wsn. *Journal of Network and Computer Applications*, 66:151–165, 2016.

[38] P. Kar, A. Roy, and S. Misra. Connectivity reestablishment in self-organizing sensor networks with dumb nodes. *ACM Transactions on Autonomous and Adaptive Systems*, 10(4):28:1–28:30, 2016.

[39] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.

[40] R. Karp. Reducibility among combinatorial problems. In R. Miller, J. Thatcher, and J. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

[41] A. Kashyap, S. Khuller, and M. Shayman. Relay Placement for Higher Order Connectivity in Wireless Sensor Networks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.

[42] S. Kim, J.-G. Ko, J. Yoon, and H. Lee. Multiple-objective metric for placing multiple base stations in wireless sensor networks. In *2007 2nd International Symposium on Wireless Pervasive Computing*. IEEE, 2007.

[43] M. Koç and I. Korpeoglu. Controlled sink mobility algorithms for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2014, 2014.

[44] M. Koç and I. Korpeoglu. Coordinated movement of multiple mobile sinks in a wireless sensor network for improved lifetime. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):1–16, 2015.

[45] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012.

[46] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *IEEE infocom*, volume 2, pages 39–44, 2002.

[47] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad hoc networks beyond unit disk graphs. *Wireless Networks*, 14(5):715–729, 2008.

[48] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester. A survey on wireless body area networks. *Wireless Networks*, 17(1):1–18, 2011.

[49] J. Li, L. L. Andrew, C. H. Foh, M. Zukerman, and H.-H. Chen. Connectivity, coverage and placement in wireless sensor networks. *Sensors*, 9(10):7664–7693, 2009.

[50] J. Li and P. Mohapatra. An analytical model for the energy hole problem in many-to-one sensor networks. In *Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd*, volume 4, pages 2721–2725, Sept 2005.

[51] M. Li, Z. Li, and A. V. Vasilakos. A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues. *Proceedings of the IEEE*, 101(12):2538–2557, 2013.

[52] N. Li and J. C. Hou. Topology control in heterogeneous wireless networks: problems and solutions. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[53] G.-H. Lin and G. Xue. Steiner tree problem with minimum number of steiner points and bounded edge-length. *Information Processing Letters*, 69(2):53–57, 1999.

[54] S. Lin, J. Zhang, G. Zhou, L. Gu, J. A. Stankovic, and T. He. Atpc: adaptive transmission power control for wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 223–236. ACM, 2006.

[55] H. Liu, A. Nayak, and I. Stojmenovic. Fault-Tolerant Algorithms/Protocols in Wireless Sensor Networks. *Guide to Wireless Sensor Networks*, pages 265–295, 2009.

[56] E. Lloyd and G. Xue. Relay Node Placement in Wireless Sensor Networks. *IEEE Trans. Computers*, 56(1):134–138, 2007.

[57] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, pages 224–235. IEEE, 2004.

[58] S. Mahmud, H. Wu, and J. Xue. Efficient energy balancing aware multiple base station deployment for wsns. In *European Conference on Wireless Sensor Networks*, pages 179–194. Springer, 2011.

[59] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *In Proceedings of ACM Wireless Sensor Networks and Applications(WSNA)*, pages 88–97, 2002.

[60] R. Mulligan and H. M. Ammari. Coverage in wireless sensor networks: A survey. *Network Protocols and Algorithms*, 2(2):27–53, 2010.

[61] E. I. Oyman and C. Ersoy. Multiple sink network design problem in large scale wireless sensor networks. In *Communications, 2004 IEEE International Conference on*, volume 6, pages 3663–3667. IEEE, 2004.

[62] L. Paradis and Q. Han. A survey of fault management in wireless sensor networks. *Journal of Network and Systems Management*, 15(2):171–190, 2007.

[63] P. Pardesi and J. Grover. Improved multiple sink placement strategy in wireless sensor networks. In *Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on*, pages 418–424. IEEE, 2015.

[64] J. Park and S. Sahni. An online heuristic for maximum lifetime routing in wireless sensor networks. *Computers, IEEE Transactions on*, 55(8):1048–1056, 2006.

[65] B. Placzek. Selective data collection in vehicular networks for traffic control applications. *Transportation Research Part C: Emerging Technologies*, 23(0):14 – 28, 2012. Data Management in Vehicular Networks.

[66] W. Y. Poe and J. B. Schmitt. Minimizing the maximum delay in wireless sensor networks by intelligent sink placement. *Distributed Computer Systems Lab University of Kaiserslautern*, 67655:1–20, 2007.

[67] L. Qiu, R. Chandra, K. Jain, and M. Mahdian. Optimizing the placement of integration points in multi-hop wireless networks. In *Proceedings of ICNP*, volume 4, 2004.

[68] V. Ranga, M. Dave, and A. K. Verma. A hybrid timer based single node failure recovery approach for wsans. *Wireless personal communications*, 77(3):2155–2182, 2014.

[69] V. Rao, P. Priyesh, and S. Kar. Adaptive transmission power protocol for heterogeneous wireless sensor networks. In *Communications (NCC), 2015 Twenty First National Conference on*, pages 1–5, Feb 2015.

[70] R. T. Rehana, R. V. Maneesha, and K. Sangeeth. Fault tolerant clustering approaches in wireless sensor network for landslide area monitoring. In *ICWN*, pages 107–113, 2008.

[71] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso. Experimental evaluation of lifetime bounds for wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 25–32, Jan 2005.

[72] H. Safa, W. El-Hajj, and H. Zoubian. Particle swarm optimization based approach to solve the multiple sink placement problem in wsns. In *Communications (ICC), 2012 IEEE International Conference on*, pages 5445–5450. IEEE, 2012.

[73] H. Safa, W. El-Hajj, and H. Zoubian. A robust topology control solution for the sink placement problem in wsns. *Journal of Network and Computer Applications*, 39:70–82, 2014.

[74] P. Santi. Topology control in wireless ad hoc and sensor networks. *ACM computing surveys (CSUR)*, 37(2):164–194, 2005.

[75] D. Sarma, H. Kumar, and A. Kar. Security threats in wireless sensor networks. In *Carnahan Conferences Security Technology, Proceedings 2006 40th Annual IEEE International*, pages 243–251. IEEE, 2006.

[76] F. Senel and M. Younis. Novel relay node placement algorithms for establishing connected topologies. *Journal of Network and Computer Applications*, 2016.

[77] R. C. Shah and J. M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350–355. IEEE, 2002.

[78] L. Sitanayah, K. N. Brown, and C. J. Sreenan. Multiple sink and relay placement in wireless sensor networks. In *Proceedings of the 1st Workshop Artificial Intelligence for Telecommunications and Sensor Networks (WAITS'12), 20th European Conference on Artificial Intelligence (ECAI'12)*, pages 18–23, 2012.

[79] L. Sitanayah, K. N. Brown, and C. J. Sreenan. A fault-tolerant relay placement algorithm for ensuring k vertex-disjoint shortest paths in wireless sensor networks. *Ad Hoc Networks*, 23:145–162, 2014.

[80] L. Sitanayah, K. N. Brown, and C. J. Sreenan. Planning the deployment of multiple sinks and relays in wireless sensor networks. *Journal of Heuristics*, 21(2):197–232, 2015.

[81] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 435–441, New York, NY, USA, 1996. ACM.

[82] Z. Sun, P. Wanga, M. Vuran, M. Al-Rodhaan, A. Al-Dhelaan, and I. Akyildiz. Bordersense: Border patrol through advanced wireless sensor networks. *Ad Hoc Networks*, 9(3):468 – 477, 2011.

[83] A. Tanenbaum and M. Steen. *Distributed systems*. Citeseer, 2002.

[84] J. Tang, B. Hao, and A. Sen. Relay Node Placement in Large Scale Wireless Sensor Networks. *Computer Communications*, 29(4):490–501, 2006.

[85] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[86] L. Wang and Y. Xiao. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mobile Networks and Applications*, 11(5):723–740, 2006.

[87] W. Wang, V. Srinivasan, and K. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, MobiCom '05, pages 270–283, New York, NY, USA, 2005. ACM.

[88] X. Wang, M. Sheng, M. Liu, D. Zhai, and Y. Zhang. Resp: A k-connected residual energy-aware topology control algorithm for ad hoc networks. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 1009–1014, April 2013.

[89] Y. Wang. Topology control for wireless sensor networks. In *Wireless Sensor Networks and Applications*, pages 113–147. Springer, 2008.

[90] Y. Wang, G. Attebury, and B. Ramamurthy. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 8(2):2–23, 2006.

[91] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *INFOCOM 2001. Twentieth annual joint conference of the IEEE computer and communications societies. Proceedings. IEEE*, volume 3, pages 1388–1397. IEEE, 2001.

[92] C. Wenjie, C. Lifeng, C. Zhanglong, and T. Shiliang. A realtime dynamic traffic control system based on wireless sensor network. In *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pages 258 – 264, june 2005.

[93] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 14–27, New York, NY, USA, 2003. ACM.

[94] C.-H. Wu and Y.-C. Chung. Heterogeneous wireless sensor network deployment and topology control based on irregular sensor model. In *Advances in Grid and Pervasive Computing*, pages 78–88. Springer, 2007.

[95] X. Xu and W. Liang. Placing optimal number of sinks in sensor networks for network lifetime maximization. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2011.

[96] T. Yan, Y. Gu, T. He, and J. A. Stankovic. Design and optimization of distributed sensing coverage in wireless sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):33, 2008.

[97] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proceedings of the IEEE Inter-*

national Conference on Computer Communication Volume 2, pages 878–890, Miami, FL, USA, 2005.

[98] X. Yin, J. Zhu, Y. Li, and Z. Wu. Mobile data gathering with time-constraints in wireless sensor networks. In *Wireless Algorithms, Systems, and Applications*, pages 696–705. Springer, 2015.

[99] Z. Yin, F. Li, M. Shen, and Y. Wang. Fault-tolerant topology for energy-harvesting heterogeneous wireless sensor networks. In *Communications (ICC), 2015 IEEE International Conference on*, pages 6761–6766, June 2015.

[100] M. Younis, S. Lee, and A. A. Abbasi. A localized algorithm for restoring internode connectivity in networks of moveable sensors. *Computers, IEEE Transactions on*, 59(12):1669–1682, 2010.

[101] M. Younis, I. F. Senturk, K. Akkaya, S. Lee, and F. Senel. Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks*, 58:254–283, 2014.

[102] W. Youssef and M. Younis. Intelligent gateways placement for reduced data latency in wireless sensor networks. In *2007 IEEE International Conference on Communications*, pages 3805–3810. IEEE, 2007.

[103] J. Zhang, Y. Liu, D. Sun, and B. Li. Prolonging the lifetime of wireless sensor networks by utilizing feedback control. *Wireless Networks*, 20(7):2095–2107, 2014.

[104] W. Zhang, G. Xue, and S. Misra. Fault-Tolerant Relay Node Placement in Wireless Sensor Networks: Problems and Algorithms. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1649–1657, 2007.

[105] S. Zhuiykov. Solid-state sensors monitoring parameters of water quality for the next generation of wireless sensor networks. *Sensors and Actuators B: Chemical*, 161(1):1 – 20, 2012.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Deniz, Fatih
Nationality: Turkish (TC)
Date and Place of Birth: 15-Mar-1985, Ankara
Marital Status: Married
Phone: +90 553 186 68 46
E-Mail: fatihdeniz@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
M.S.	METU, Computer Engineering	2010
B.S.	Bilkent University, Computer Engineering	2007

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2012 - ...	Central Bank of of Turkey	IT Security Specialist
2011 - 2012	BTK	IT Specialist
2007 - 2011	TUBİTAK / BİLGEM / İLTAREN	Researcher

PUBLICATIONS

International Journal Publications

Fatih Deniz, Hakki Bagci, Ibrahim Korpeoglu, and Adnan Yazıcı. An adaptive, energy-aware and distributed fault-tolerant topology-control algorithm for heterogeneous wireless sensor networks. Ad Hoc Networks, 44:104–117, July 2016.

Fatih Deniz, Nedim Alpdemir, Ahmet Kara, and Halit Oğuztüzün. Supporting dynamic simulations with Simulation Modeling Architecture (SiMA): a Discrete Event System Specification-based modeling and simulation framework. Simulation, vol. 88, no. 6, pages 707-730, June 2012.

International Conference Publications

Fatih Deniz, Hakki Bagci, Ibrahim Korpeoglu, Ahmet Oguz Akyuz, and Adnan Yazici. Energy-Efficient and Fault-Tolerant Supernode Placement in Heterogeneous Wireless Sensor Networks. To be submitted.

Fatih Deniz, Ahmet Kara, Nedim Alpdemir and Halit Oguztuzun. Variable Structure and Dynamism Extensions to SiMA, A DEVS Based Modeling and Simulation Framework. In Proceedings of Summer Computer Simulation Conference (SCSC'09), pages 117–124, 2009.

Ahmet Kara, Fatih Deniz, Doruk Bozagaç and Nedim Alpdemir. 2009. Simulation Modeling Architecture (SiMA), A DEVS Based Modeling and Simulation Framework. In Proceedings of Summer Computer Simulation Conference (SCSC'09), pages 315–321.