UNIVERSIDAD POLITÉCNICA DE MADRID

Escuela Técnica Superior
de
Ingeniería y Sistemas de Telecomunicación

Contributions to the Resilience Management in the Internet of Things

TESIS DOCTORAL

Yuanjiang Huang

Master in Communication and Information System

2015

Centro de Investigación en Tecnologías de Software y Sistemas Multimedia para la Sostenibilidad

Escuela Técnica Superior de
Ingeniería y Sistemas de Telecomunicación

Contributions to the Resilience Management in the Internet of Things

Doctoral Thesis

Yuanjiang Huang

Master in Communication and Information System

Supervisors:
Prof. PhD. José-Fernán Martínez Ortega
Prof. PhD. Juana Sendra Pons

Universidad Politécnica de Madrid

2015

DOCTORADO EN INGENIERÍA DE SISTEMAS Y SERVICIOS PARA LA SOCIEDAD DE LA INFORMACIÓN

Tesis Doctoral		
Título	Contributions to the Resilience Management in the Internet of Things	
Autor	Yuanjiang Huang	
Co-Director	Dr. José-Fernán Martínez Ortega	V^{o}B^{o}.
Co-Directora	Dra. Juana Sendra Pons	V^{o}B^{o}.
Tribunal		
Presidente	Dr. Juan Ramón Velasco Pérez	
Secretaria	Dra. Lourdes López Santidrián	
Vocal	Dr. Lukasz Stanislaw Kulas	
Vocal	Dr. Juan Carlos Dueñas López	
Vocal	Dr. Carlos García Rubio	
Suplente	Dr. Luigi Glielmo	
Suplente	Dr. Rodolfo Haber Guerra	
Lugar y fecha de lectura	E.T.S.I. y Sistemas de Telecomunicación (U.P.M.)	
Calificación		

El Presidente La secretaria Los vocales

Tesis Doctoral para la obtención del título de Doctor
por la Universidad Politécnica de Madrid

2015

To my beloved family.

Contents

Acknowledgements **i**

Abstract **iii**

Resumen **v**

List of Figures **ix**

List of Tables **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem statement and objective 3

1.2.1 Basic concepts 3

1.2.2 Objective and methodology 7

1.3 Contributions 8

1.4 Thesis structure 9

2 Related Works **11**

2.1 Network design flow 11

2.2 Network model 13

2.2.1 Heterogeneous and homogenous network 13

2.2.2 Transmission model 13

2.2.3 Failure models 15

2.3 Topology control 17

2.3.1 Topology control in different deployment stages 17

2.3.2 Pre-deployment 18

2.3.3 Post-deployment 22

2.3.4 Re-deployment 26

2.4 Multi-path routing protocol 28

2.4.1 Complete disjointed and brained path 29

2.4.2 Multi-path searching algorithm 30

2.5 NP-completeness and NP-hard problems in resilient IoT 32
2.6 Adopted models 34
2.7 Summary 35

3 Connectivity Analysis Using a Probabilistic Model 37
3.1 Network model 37
3.2 Homogenous node deployment 38
3.2.1 Network connection probability analysis 38
3.2.2 Validation 44
3.3 Heterogeneous node deployment 45
3.3.1 Cox process model 45
3.3.2 Network connection probability analysis 46
3.3.3 Numerical simulations 46
3.3.4 Choose distribution parameter 49
3.4 Summary 50

4 Maintain the IoT Resilience Using Fuzzy-logic Control 51
4.1 Fuzzy-logic control theory 51
4.2 Fuzzy-logic control based on learning algorithm 53
4.2.1 LFTC learning data set 56
4.2.2 LFTC algorithm complexity 57
4.2.3 Controller parameters 57
4.2.4 LFTC protocol 58
4.3 Fuzzy-logic control based on if-then rules 59
4.4 Matlab-based simulation tool 60
4.5 System validation 61
4.5.1 Control system analysis 62
4.5.2 Representative algorithms 63
4.5.3 Network model and configurations 65
4.6 Simulation results and analysis 66
4.6.1 Resultant topology 66
4.6.2 Average node degree 66
4.6.3 Topology control on initial network 68
4.6.4 Topology control on network with random failures 70
4.7 Summary 72

5 A Lightweight Self-adaptive Control System Design and Optimization 73
5.1 Two-loop controller 73
5.2 Control system parameter optimization 78
5.2.1 A lightweight control system design 78
5.2.2 Parameters to optimize 79

5.2.3 Objective function 80
5.3 Optimization algorithms 82
5.4 Simulations and validation 84
5.5 Summary 88

6 Evaluation via Testbed **91**
6.1 DEMANES 92
6.2 Self-adaptive software 92
6.2.1 Component-based design for DEMANES middleware 93
6.2.2 Self-adaptive software techniques 94
6.3 Middleware implementation 98
6.3.1 Control system implementation 98
6.3.2 Service-oriented architecture 102
6.4 Experimental results 103
6.4.1 Experiment setup 104
6.4.2 Indoor test 105
6.4.3 Outdoor test 106
6.5 Summary 109

7 Conclusions and Future Works **111**
7.1 Conclusions 111
7.2 Future works 113
7.2.1 Improve models and evaluation methods 114
7.2.2 Extend current models 115

Bibliography **117**

List of Author's Publications **133**

Acknowledgements

This thesis would not have been possible without the support of many people. Apart from the knowledge I have acquired in the past four years, it has been a lot of fun to work with the people in the "Next Generation Networks and Services" group at UPM.

First and foremost, I want to thank Prof. José-Fernán Martínez Ortega and Prof. Juana Sendra Pons, who have had significant impact on my research, as my supervisors and as my friends. Prof. José-Fernán encouraged me during difficult times, and inspired me when writing scientific papers. Most importantly, Prof. José-Fernán provided me the perfect balance of guidance and freedom during the past years. Prof. Juana Sendra helped me in developing the mathematical models, and giving me valuable comments and criticisms on my manuscripts. I would also like to thank both of them for spending countless time to read this thesis. Without their help and support, I would not be where I am today.

I am also deeply grateful to Prof. Vicente Hernández Díaz, Néstor Lucas Martínez, and Raúl Mario del Toro Matamoros, who contributed to ideas, insights and useful feedback. They have shown me how to understand and develop the software for sensor networks. It was my pleasure to discuss with them, and learn lots of techniques from them.

It is impossible to acknowledge all people. However, I wish to express my gratitude to the past and current colleagues in the group. Special thanks are due to Pedro Castillejo for providing useful advices and always being nice to talk to, and Jesus Rodríguez for his help in improving my written English. Also, I would like to thank: Prof. Lourdes López Santidrián, Prof. Martina Eckert, Alexandra Cuerva, David Gómez, Esther Moreno, José Antonio Sánchez Alcón, Carlos Estévez Novo, and my Chinese colleagues Xin Li, Ning Li, Xin Yuan.

Finally, no one deserves my thanks more than my family, for their love and support. Special thanks also are due to Juan Liao. You are the reward for my PhD career. Although it has been difficult for us being separated so long, during all this time you have always been by my side. I could never have done this without you.

Abstract

The advent of the Internet of Things (IoT) enables a tremendous number of applications, such as forest monitoring, disaster management, home automation, factory automation, smart city, etc. However, various kinds of unexpected disturbances may cause node failure in the IoT, for example battery depletion, software/hardware malfunction issues and malicious attacks. So, it can be considered that the IoT is prone to failure. The ability of the network to recover from unexpected internal and external failures is known as "resilience" of the network. Resilience usually serves as an important non-functional requirement when designing IoT, which can further be broken down into "self-*" properties, such as self-adaptive, self-healing, self-configuring, self-optimization, etc.

One of the consequences that node failure brings to the IoT is that some nodes may be disconnected from others, such that they are not capable of providing continuous services for other nodes, networks, and applications. In this sense, the main objective of this dissertation focuses on the IoT connectivity problem. A network is regarded as connected if any pair of different nodes can communicate with each other either directly or via a limited number of intermediate nodes. More specifically, this thesis focuses on the development of models for analysis and management of resilience, implemented through the Wireless Sensor Networks (WSNs), which is a challenging task. On the one hand, unlike other conventional network devices, nodes in the IoT are more likely to be disconnected from each other due to their deployment in a hostile or isolated environment. On the other hand, nodes are resource-constrained in terms of limited processing capability, storage and battery capacity, which requires that the design of the resilience management for IoT has to be lightweight, distributed and energy-efficient.

In this context, the thesis presents self-adaptive techniques for IoT, with the aim of making the IoT resilient against node failures from the network topology control point of view. The fuzzy-logic and proportional-integral-derivative (PID) control techniques are leveraged to improve the network connectivity of the IoT in response to node failures, meanwhile taking into consideration that energy consumption must be preserved as much as possible. The control algorithm itself is designed to be distributed, because the centralized approaches are usually not feasible in large scale IoT deployments. The thesis involves various aspects concerning network connectivity, including: creation and analysis of mathematical models describing the network, proposing self-adaptive control systems in response to node failures, control system parameter optimization, implementation using the software engineering approach, and evaluation in a real application.

This thesis also justifies the relations between the "node degree" (the number of neighbor(s) of a node) and network connectivity through mathematic analysis, and proves the effectiveness of various types of controllers that can adjust power transmission of the IoT nodes in response to node failures. The controllers also take into consideration the energy consumption as part of the control goals. The evaluation is performed and comparison is made with other representative algorithms. The simulation results show that the proposals in this thesis can tolerate more random node failures and save more energy when compared with those representative algorithms. Additionally, the simulations demonstrate that the use of the bio-inspired algorithms allows optimizing the parameters of the controller. With respect to the implementation in a real system, the programming model called OSGi (Open Service Gateway Initiative) is integrated with the proposals in order to create a self-adaptive middleware, especially reconfiguring the software components at runtime when failures occur.

The outcomes of this thesis contribute to theoretic research and practical applications of resilient topology control for large and distributed networks. The presented controller designs and optimization algorithms can be viewed as novel trials of the control and optimization techniques for the coming era of the IoT. The contributions of this thesis can be summarized as follows: (1) Mathematically, the fault-tolerant probability of a large-scale stochastic network is analyzed. It is studied how the probability of network connectivity depends on the communication range of the nodes, and what is the minimum number of neighbors to be added for network re-connection. (2) A fuzzy-logic control system is proposed, which obtains the desired node degree and in turn maintains the network connectivity when it is subject to node failures. There are different types of fuzzy-logic controllers evaluated by simulations, and the results demonstrate the improvement of fault-tolerant capability as compared to some other representative algorithms. (3) A simpler but more applicable approach, the two-loop control system is further investigated, and its control parameters are optimized by using some heuristic algorithms such as Cross Entropy (CE), Particle Swarm Optimization (PSO), and Differential Evolution (DE). (4) Most of the designs are evaluated by means of simulations, but part of the proposals are implemented and tested in a real-world application by combining the self-adaptive software technique and the control algorithms which are presented in this thesis.

Keywords: Internet of Things, resilience management, fault-tolerant, topology control, control algorithm, optimization theory, self-adaptive software.

Resumen

El auge del "Internet de las Cosas" (IoT, "Internet of Things") y sus tecnologías asociadas han permitido su aplicación en diversos dominios de la aplicación, entre los que se encuentran la monitorización de ecosistemas forestales, la gestión de catástrofes y emergencias, la domótica, la automatización industrial, los servicios para ciudades inteligentes, la eficiencia energética de edificios, la detección de intrusos, la gestión de desastres y emergencias o la monitorización de señales corporales, entre muchas otras. La desventaja de una red IoT es que una vez desplegada, ésta queda desatendida, es decir queda sujeta, entre otras cosas, a condiciones climáticas cambiantes y expuestas a catástrofes naturales, fallos de software o hardware, o ataques maliciosos de terceros, por lo que se puede considerar que dichas redes son propensas a fallos. El principal requisito de los nodos constituyentes de una red IoT es que estos deben ser capaces de seguir funcionando a pesar de sufrir errores en el propio sistema. La capacidad de la red para recuperarse ante fallos internos y externos inesperados es lo que se conoce actualmente como "Resiliencia" de la red. Por tanto, a la hora de diseñar y desplegar aplicaciones o servicios para IoT, se espera que la red sea tolerante a fallos, que sea auto-configurable, auto-adaptable, auto-optimizable con respecto a nuevas condiciones que puedan aparecer durante su ejecución.

Esto lleva al análisis de un problema fundamental en el estudio de las redes IoT, el problema de la "Conectividad". Se dice que una red está conectada si todo par de nodos en la red son capaces de encontrar al menos un camino de comunicación entre ambos. Sin embargo, la red puede desconectarse debido a varias razones, como que se agote la batería, que un nodo sea destruido, etc. Por tanto, se hace necesario gestionar la resiliencia de la red con el objeto de mantener la conectividad entre sus nodos, de tal manera que cada nodo IoT sea capaz de proveer servicios continuos, a otros nodos, a otras redes o, a otros servicios y aplicaciones.

En este contexto, el objetivo principal de esta tesis doctoral se centra en el estudio del problema de conectividad IoT, más concretamente en el desarrollo de modelos para el análisis y gestión de la Resiliencia, llevado a la práctica a través de las redes WSN, con el fin de mejorar la capacidad la tolerancia a fallos de los nodos que componen la red. Este reto se aborda teniendo en cuenta dos enfoques distintos, por una parte, a diferencia de otro tipo de redes de dispositivos convencionales, los nodos en una red IoT son propensos a perder la conexión, debido a que se despliegan en entornos aislados, o en entornos con condiciones extremas; por otra parte, los nodos suelen ser recursos con bajas capacidades en términos de procesamiento, almacenamiento y batería, entre otros, por lo que requiere que el diseño de la gestión de su resiliencia sea ligero, distribuido y energéticamente eficiente.

En este sentido, esta tesis desarrolla técnicas auto-adaptativas que permiten a una red IoT, desde la perspectiva del control de su topología, ser resiliente ante fallos en sus nodos. Para ello, se utilizan técnicas basadas en lógica difusa y técnicas de control proporcional, integral y derivativa (PID - "proportional-integral-derivative"), con el objeto de mejorar la conectividad de la red, teniendo en cuenta que el consumo de energía debe preservarse tanto como sea posible. De igual manera, se ha tenido en cuenta que el algoritmo de control debe ser distribuido debido a que, en general, los enfoques centralizados no suelen ser factibles a despliegues a gran escala. El presente trabajo de tesis implica varios retos que conciernen a la conectividad de red, entre los que se incluyen: la creación y el análisis de modelos matemáticos que describan la red, una propuesta de sistema de control auto-adaptativo en respuesta a fallos en los nodos, la optimización de los parámetros del sistema de control, la validación mediante una implementación siguiendo un enfoque de ingeniería del software y finalmente la evaluación en una aplicación real.

Atendiendo a los retos anteriormente mencionados, el presente trabajo justifica, mediante una análisis matemático, la relación existente entre el "grado de un nodo" (definido como el número de nodos en la vecindad del nodo en cuestión) y la conectividad de la red, y prueba la eficacia de varios tipos de controladores que permiten ajustar la potencia de trasmisión de los nodos de red en respuesta a eventuales fallos, teniendo en cuenta el consumo de energía como parte de los objetivos de control. Así mismo, este trabajo realiza una evaluación y comparación con otros algoritmos representativos; en donde se demuestra que el enfoque desarrollado es más tolerante a fallos aleatorios en los nodos de la red, así como en su eficiencia energética. Adicionalmente, el uso de algoritmos bioinspirados ha permitido la optimización de los parámetros de control de redes dinámicas de gran tamaño. Con respecto a la implementación en un sistema real, se han integrado las propuestas de esta tesis en un modelo de programación OSGi ("Open Services Gateway Initiative") con el objeto de crear un middleware auto-adaptativo que mejore la gestión de la resiliencia, especialmente la reconfiguración en tiempo de ejecución de componentes software cuando se ha producido un fallo.

Como conclusión, los resultados de esta tesis doctoral contribuyen a la investigación teórica y, a la aplicación práctica del control resiliente de la topología en redes distribuidas de gran tamaño. Los diseños y algoritmos presentados pueden ser vistos como una prueba novedosa de algunas técnicas para la próxima era de IoT. A continuación, se enuncian de forma resumida las principales contribuciones de esta tesis: (1) Se han analizado matemáticamente propiedades relacionadas con la conectividad de la red. Se estudia, por ejemplo, cómo varía la probabilidad de conexión de la red al modificar el alcance de comunicación de los nodos, así como cuál es el mínimo número de nodos que hay que añadir al sistema desconectado para su re-conexión. (2) Se han propuesto sistemas de control basados en lógica difusa para alcanzar el grado de los nodos deseado, manteniendo la conectividad completa de la red. Se han evaluado diferentes tipos de controladores basados en lógica difusa mediante simulaciones, y los resultados se han comparado con otros algoritmos representativos. (3) Se ha investigado más a fondo, dando un enfoque más simple y aplicable, el sistema de control de doble bucle, y sus parámetros de control se han optimizado empleando algoritmos heurísticos como el método de la entropía cruzada (CE, "Cross Entropy"), la optimización por enjambre de partículas (PSO, "Particle Swarm Optimization"), y la evolución diferencial (DE, "Differential Evolution"). (4) Se han evaluado mediante simulación, la mayoría de los diseños aquí presentados; además, parte de los trabajos se han implementado y validado en una aplicación real combinando técnicas de software auto-adaptativo, como por ejemplo las de una arquitectura orientada a servicios (SOA, "Service-Oriented Architecture").

Palabras clave: Internet de las Cosas, resiliencia, tolerancia a fallos, control topológico, algoritmos de control, teoría de optimización, software auto-adaptativo.

List of Figures

1.1 Network topology example. (1) $|V| = 7$, $|E| = 11$. (2) It is non-negative weighted edge undirected connected graph. (3) Four nodes 6, 3, 7 and 4 are neighbors of node 5, so $d(v_5) = 4$. (4) Network degree $\delta = 2$. (5) It is 2-connected and 2-edge-connected. 3

2.1 Resilient IoT design flow. 12
2.2 Regular and irregular radio model. 14
2.3 (a) Symmetric link: Node 1 can communicate with node 2 and vice versa. (b) Asymmetric link: node 1 can send data to node 2, but node 2 cannot send data to node 1; (c) Non-omnidirectional radio. 15
2.4 Topology control: adjust the transmission range of node 3 from R1 to R2 in order to improve connectivity. 18
2.5 Topology control: new node 6 is added to recover connection. 19
2.6 Construct new $2 - \textit{connected}$ network H from existing $2 - \textit{connected}$ network G. 22
2.7 Disjoint paths and brained paths. (1, 2, 6) and (1, 3, 5, 6) are disjoint-paths. (1, 3, 4, 6) and (1, 3, 5, 6) are brained paths. 29
2.8 Construct disjoint paths [49]. 31

3.1 (a) Disk communication model and (b) Irregular communication model. 38
3.2 Network connection probability $P(t)$ when $n = 500, 1000, 10000, 100000$. 40
3.3 Network connection probability function $P(t)$, and corresponding $P'(t)$ and $P''(t)$ (up); values t_1 and t_2 for a given ε and $n = 500$ (down). . . . 41
3.4 (a) Errors of Theorem 3.2.4; (b) Errors of Theorem 3.2.6. 44
3.5 Cumulative distribution function (CDF) of P_k when $\sigma = 1, 2, 3$, n = 500, and $k = 1$. 47
3.6 Cumulative distribution function (CDF) of P_k when $\sigma = 1, 2, 3$ and n = 500, $k = 2$. 48
3.7 $k - \textit{connected}$ probability P when $\sigma = 2$, and $k = 1, 2, 3, 4$. 48
3.8 Connection probability P when $\sigma = 1, 2, 3$ and $k = 1$. 48
3.9 Required σ for 90% connection probability when $\mu = 10$. 49

4.1 Fuzzy-logic control system. 52
4.2 Learning-based Fuzzy-logic Topology Control system (LFTC). 53
4.3 Alternative: Learning-based Fuzzy-logic Topology Control system (LFTC). 55
4.4 Rules-based Fuzzy-logic Topology Control (RFTC). 60
4.5 Simplified node model in Matlab-based simulator. 61
4.6 Average node degree with different K ($Prob_0 = 0.8$, n = 60, k = 3). . . . 62
4.7 Average node degree with different $Prob_0$ (K = 0.02, n = 60, k = 3). . . 62
4.8 Example of resultant topology (n = 65, k = 3). (a) NONE; (b) k-Neighbor; (c) FLSS; (d) LTRT; (e) FTC. 67
4.9 Average Node Degree with Different k. 69
4.10 Average Communication Range (k = 3). 69
4.11 Maximum Communication Range (k = 3). 70
4.12 Energy Expended Ratio (EER) (k = 3). 70
4.13 Average Node Degree for Different Failure Probabilities (k = 3, n = 60). 71

5.1 Reconfiguration loops for a self-adaptive communication range. 74
5.2 Matlab-based simulation tool. 76
5.3 Simplified control system design. 79
5.4 Network topology. 85
5.5 $W_e = W_p = 0.5$, mean of J at each iteration. 86
5.6 $W_e = 0.2, W_p = 0.8$, mean values of J at each iteration. 87
5.7 Standard deviations for CE. 87
5.8 The comparison among CE, PSO, and DE. 88
5.9 The comparison among CE, PSO, and DE in asynchronous network. . . 89

6.1 Feedback control for the self-adaptive software. 96
6.2 Self-adaptive architecture. 97
6.3 The control system implementation in DEMANES middleware. 99
6.4 The reconfiguration class diagram of DEMANES middleware. 100
6.5 Three-handshake neighbor discovery protocol. 101
6.6 OSGi architecture [109]. 103
6.7 Indoor deployment environment: the south campus at UPM. 104
6.8 Indoor test: the number of nodes connected to the BS. 105
6.9 Outdoor deployment environment: the south campus at UPM. 106
6.10 The outdoor locations of the Sun SPOT sensors. 107
6.11 Outdoor test: the number of nodes connected to the BS. 108
6.12 Outdoor test: the number of nodes connected to the BS when failures occur. 108

List of Tables

2.1 NP-completeness and NP-hard problems in resilient IoT. 32
2.2 The models adopted in the thesis. 34

3.1 Connection probability with different size of network. 45

4.1 Fuzzy-Logic If-Then Rules. 60

5.1 Comparison of received packets at BS. 77

6.1 Self-adaptive software techniques. 95

Chapter 1

Introduction

In this chapter, the motivations and objectives of the thesis are introduced, and the contributions for the studied problems are highlighted. This chapter ends with outlining the structure of the thesis.

1.1 Motivation

"Internet of Things" (IoT) [9][30] nowadays promotes a tremendous number of novel applications [154], and it is boosting the world economy and improving people's lives. IoT is viewed as the evolution and revolution of the current Internet network, because it is not only a network of computers and mobile phones, but also a network made up by devices of all types and sizes that can be imagined. All of those devices are called "Things" (or, "nodes" in the context of networks). Nowadays, the IoT becomes one of the most active researches in academia and industry. For example, research and application in forest monitoring, disaster management, home automation, factory automation, smart city, medical and health-care, etc., are on the rise. European Research Cluster (IERC) on the Internet of Things, defined the IoT as follows [145]: "Internet of things (IoT): A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies." The IoT is inherently multi-disciplinary. Enabling technologies cover a huge scope of disciplines, e.g. Wireless Sensor Networks (WSNs) [152][4], RFID, M2M, mobile Internet, 2G/3G/4G, artificial intelligent, semantic data integration, semantic search, IPv6, etc. In particular, WSNs technology, which is the basis of the IoT, has been studied long time ago.

The IoT puts a significant amount of emphasis on network connectivity. The goal of the IoT is to enable things to be connected anytime, anywhere, with anything and anyone ideally using any path/network and any service [131]. This goal motivated the study of this thesis, because the IoT is very vulnerable to connection loss due to various reasons. The IoT nodes are usually battery-powered, deployed either randomly or according to a predefined statistical distribution [50] over an environment which is possibly prone to unexpected failures and malicious attacks. For example, in environment surveillance, sensor nodes are exposed to an unsafe situation, making nodes likely to suffer various kinds of damage. On the other hand, the node itself is very vulnerable due to unreliable wireless connection and resource-constrained features, such as limited transmission power, computing ability, storage space, etc.

The network resilience is defined as the network ability to provide and maintain an acceptable level of service in the face of various faults and challenges for normal operations [126]. [49] defines resilience as the probability of at least one alternate path available within the interval T, given that at least one node on the primary path has failed. Resilience apparently is not only the desirable requirement for the IoT, but also for any other networks and control systems. For instance, resilience is commonly found in natural systems, and is a hallmark of the biological world [21]. Many analysts in engineering do not differentiate resilience from robustness and survivability [130]. Robustness and survivability tend to be used interchangeably with resilience, even though in practice these properties are quite different [21]. Resilience, robustness and survivability are system-wide properties [73]. Resilience can also be described as the ability to recover to the original status. The goal of resilience is to build fault-tolerant networks. By contrast, robustness stresses the ability to resist attacks, but does not imply an ability to restore from failure [21]. If the system is robust, it means that it is really difficult to degrade under various attacks. As for survivability, it indicates that the system is really difficult to impair completely.

In ecology, resilience can be measured by looking at the time that the system takes to return to its equilibrium state after a perturbation [73]. But it is difficult to provide a clear quantitative assessment of network resilience to node failure [95]. The conventional evaluation is the number of failures the network can sustain before network disconnection, or the probability to keep the network connected [104]. Briefly, if the network can tolerate at most $k-1$ node failures and the resulting network is still connected, the network is called $k-connected$ network. If the network is $k-connected$, it also means that there are at least $k-disjoint$ paths from any given two distinct nodes. Therefore, if one path fails, there are still $k-1$ alternative paths available. On the other hand, resilience of networks also depends on the link quality. Reliable links can make better connectivity quality.

In a nutshell, the particular characteristics of the IoT under hostile environments

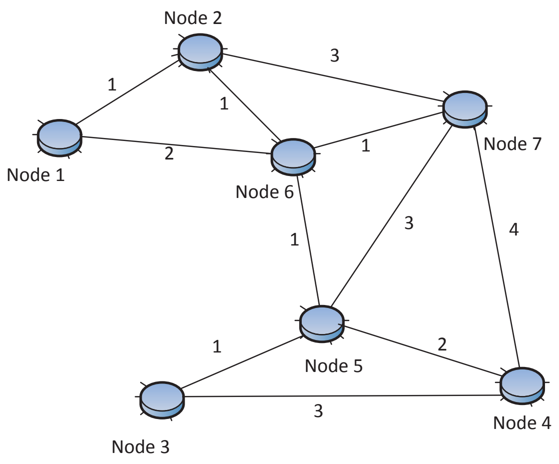


Figure 1.1: *Network topology example. (1) $|V| = 7$, $|E| = 11$. (2) It is non-negative weighted edge undirected connected graph. (3) Four nodes 6, 3, 7 and 4 are neighbors of node 5, so $d(v_5) = 4$. (4) Network degree $\delta = 2$. (5) It is 2-connected and 2-edge-connected.*

raise the needs to demonstrate self-adaptive behaviors so that the IoT will tolerate failures, in order to maintain the network at a satisfactory level of service when failures happen, which motivated the study of the thesis. This thesis leverages advances to improve resilience of IoT in terms of network connectivity. Applications will benefit from the resilient IoT as it provides favorable conditions where continuous services are more likely to happen.

1.2 Problem statement and objective

In order to define the problems and the objectives of this work, there are some concepts have to be introduced first. This section introduces the way to present network topology and the definitions that will be used in this thesis, and at the same time presents significant findings related to a $k-connected$ network. For more graph terminologies, readers are referred to the book [3]. Throughout this thesis, the terms "network" and "graph", as well as "thing" and "node", are used interchangeably.

1.2.1 Basic concepts

Node Degree (ND) is the key concept of this research. In brief, ND indicates the number of neighbor(s) an IoT node has. Higher ND implies better connectivity for the whole network as it supplies more alternative paths to other nodes. This intuition will be justified in Chapter 3. The ND will be controlled to improve/maintain network connectivity in Chapter 4 and Chapter 5. In this section, the definition of ND will be formulated.

Formally, as shown in Figure 1.1, the IoT can be described as a simple, non-negative weighted edge, undirected (or directed) graph $G = (V, E)$ in a 2-dimensional area, where V and E are the set of vertexes and edges, representing sensor nodes and links respectively. Each edge has a non-negative value called weight, representing the cost of edge, e.g. energy, distance or delay, etc. The cardinality of vertex (sensor node) and edge (link) in the graph is denoted by $|E|$ and $|V|$. Let $r_{c,i}$ denote the communication range and $r_{s,i}$ the sensing range of vertex i. $r_{c,i}$ is the longest distance the radio of a sensor can reach, and $r_{s,i}$ is the longest distance a sensor can sense environment phenomena (e.g. temperature), and usually $r_{c,i} \geq r_{s,i}$. The sensing and communication range of sensors are represented by disks, including their boundaries. The communication range and sensing range can be simply written as r_c and r_s in case that all vertexes have the same communication and sensing range.

For any two distinct vertices $v_i, v_j \in V$, the vertex v_i can connect to v_j if and only if the Euclidean distance $|v_i - v_j| \leq r_{c,i}$.

Definition 1.2.1. *Neighbor set* : The neighbor set of v_i is denoted as $N(v_i)$ and defined as any nodes within the communication range of v_i, namely:

$$N(v_i) = \{v_j : |v_i - v_j| \leq r_{c,i}, v_j \in V\}$$

The number of neighbors of v_i is called the *degree of* v_i, denoted as $d(v_i)$.

Definition 1.2.2. *Graph degree* : The graph degree is defined as $\delta = min(d(v_i))$ for all $v_i \in V$.

For example, in Figure 1.1 node 5 has degree 4 and the graph degree is 2. A point p is covered by the sensor node v only when the Euclidean distance is less than the sensing range, that is $|p - v| \leq r_{s,v}$.

In fact, the communication and sensing ranges are not necessarily a disk, and in reality they are not the hard boundary, e.g. the sensing range could be defined as the threshold of false alarm rate. Therefore, the statistical nature of sensor network applications and the environments can be incorporated in the definitions of sensing range and communication range, e.g. [134]. This will be discussed in Section 2.2.

Based upon the description above, the node degree (ND) for a node in IoT is defined as:

Definition 1.2.3. *Node degree (ND)*: number of direct neighbors for a node in the undirected graph.

Note that it has to be an undirected graph because in practice the communication between two nodes normally requires to be symmetrical. In addition, ND is directly related to communication range of itself and other nodes close to it in the wireless communication environments.

$k-$connected and $k-$coverage network

In this section, the definitions about network connectivity and coverage are introduced.

Definition 1.2.4. *Connected network*: For any two distinct vertexes, if they can communicate with each other either directly or via a limited number of intermediate vertexes, then the graph is called connected graph.

Definition 1.2.5. *k-connected network*: Given integer $k \geq 1$, if the graph is still connected after removing no more than $k-1$ vertex, then G is called $k-vertex-connectivity$ graph, or $k-vertex-connected$ graph, usually simply denoted as $k-connectivity$ graph or $k-connected$ graph. The value k is called *connectivity*. The vertex connectivity κ of the graph is the maximum k such that the graph is $k-connected$ graph.

Definition 1.2.6. *k-edge-connected*: Given integer $k \geq 1$, if the graph is still connected after removing no more than $k-1$ edge, then G is named $k-edge-connectivity$ graph, or $k-edge-connected$ graph. The vertex connectivity κ' of graph is the maximum k such that the graph is $k-edge-connected$ graph.

For example, in Figure 1.1, if two nodes 2 and 6 are removed, the resulting network is disconnected because the node 1 will be isolated; but removing only one node it is not possible to disconnect the network. The relation between vertex and edge connectivity is given by the following formula (1.1), which is called Menger's theorem [3][57]:

$$\kappa \leq \kappa' \leq \delta \leq \frac{2|E|}{|V|} \tag{1.1}$$

Apparently, the connectivity k helps to evaluate the IoT fault tolerance capability. The higher κ or κ' is, the more resilient the IoT is. The upper bound of connectivity κ can be achieved by Harary graphs. Lower bound of κ is 0 or $|E| - \frac{(|V|-1)(|V|-2)}{2}$ [57]. The connectivity in most cases is a desirable feature, but in few cases it is in fact an unwanted feature, e.g. because of easier virus propagation [135]. On the other hand, a high degree also helps to detect failures. [96] shows that optimal error detection decreases exponentially with the increase of degree. To some extent, improving network resilience means increasing, maintaining or constructing the required κ or κ'. In this thesis, when the network is $k-connected$, it means $\kappa-connected$; if the network is $k-edge-connected$, it means $\kappa'-edge-connected$.

Definition 1.2.7. *k-coverage network*: the network is called $k-coverage$ network where k is the biggest positive integer such that any point in the deployment area is covered by at least k distinct sensor nodes via the sensing range.

Partially connected network

The former definitions require that the entire network is $k-connected$ or $k-coverage$. However, as the IoT is characterized by high density, only a small number of nodes is needed for connectivity or coverage in order to save energy and prolong the IoT lifetime. As a result, maintaining a whole network connected is not only quite difficult but also may be unnecessary. If the goal is not to maintain the whole network connected, it is called *partially connected network*.

Definition 1.2.8. *Partial $k-connected$ network*: If the original network is damaged, new nodes are added to the network. The goal is not to recover the entire new network to be $k-connected$, but to maintain the original nodes to be $k-connected$.

Partial $k-connected$ weakens the final objective but is still useful in the case that only original nodes serve an important purpose, while the added nodes only are used to maintain connectivity. On the other hand, IoT has many redundant nodes, so maybe only part of them is required to be connected, e.g. the giant component should be connected as large as possible.

Definition 1.2.9. *Giant component threshold*: there exists a critical value such that whenever the fraction of removed nodes (or links, depending on the context) is lower than P_c, the network still holds a giant component almost for sure; whereas, whenever the fraction of removed nodes (or links) is greater than P_c the network is almost sure not to have a giant component anymore.

For random failures in large random deployed networks, P_c is given by equation (1.2) [98]:

$$P_c = 1 - \frac{< k >}{< k^2 > - < k >} \tag{1.2}$$

where $< k^j >= \sum_{k=0}^{\infty} k^j p_k$ (where $j = 1, 2$) and p_k is the probability distribution for all k, e.g. Poisson distribution $p_k = e^{-z}\frac{z^k}{k!}$. Notice that this conclusion is suitable for any degree distribution, e.g. Poisson distribution, continuous and discrete power law. More details about the results under other two conditions are available in [98]: the number of nodes tends to infinite, or the number of nodes is a given large number N.

In case that only part of nodes are required to be connected (rather than all of them), then studying the connectivity problem of the subset of nodes would be interesting, and it is associated with the concept minimal connected dominating set (CDS).

Definition 1.2.10. *Minimal connected dominating set (CDS)*: Let the connected dominating set D be a sub-graph of the graph G, such that: (1) D is a connected sub-graph of G, (2) any vertex in G either belongs to D or is adjacent to a vertex of D. One graph could have more than one D. The minimal CDS is the one that has minimal number of vertices than any other connected dominating sets.

If there is a connected dominating set with size less than a given value, it is a NP-completeness problem. So far, the best approximations for finding the minimal dominating set are $O(\log |V|) - approximation$ [53] and $O(\log \Delta) - approximation$ distributed algorithm [44], where Δ is the maximal degree of graph. There are several papers, such as [129][111], present the topology control algorithms through constructing an optimal-size CDS to achieve a trade-off between the network lifetime and the network coverage.

1.2.2 Objective and methodology

From the network topology point of view, if the network is $k - connected$, the higher the k value, the more resilient is the network. If it does not strictly require the whole network being $k - connected$ (e.g. there are too many redundant nodes), then a partial connected network is still acceptable. The $k - connected$ network concept, together with ND, leads to the general objective: *improving the connectivity of the IoT in the presence of node failures by means of managing the ND for each node.* The objective can further be described by several sub-objectives:

- Identifying important trends and advances in the research of resilient IoT.
- Determining the correlations between network resilience and topology control.
- Proposing novel designs to make the IoT resilient against failures.
- Evaluating the effectiveness of techniques used on topology control.

Furthermore, the objectives will be studied by using the following methodologies:

- Studying recent publications from relevant conferences and journals.
- Formulating the problem, then establish and derive the mathematic models that can describe the relations among ND, network connectivity, and fault-tolerance capability.
- Leveraging existing tools, models, and algorithms that help to make decisions on modifying ND at runtime whenever it is necessary.
- Optimizing the design to improve overall performance.
- Evaluating the proposals by means of simulation.
- Implementing and evaluating the proposals in a real system.

Each of the objectives will be tackled in separate chapters by using different techniques. Due to the fact that network resilience is inherently interdisciplinary, there are several disciplines involved during the development of this thesis, such as probability theory (Chapter 3), control theory (Chapter 4), optimization theory (Chapter 5), and soft engineering (Chapter 6). Each chapter is dependent on others and highly correlated. The theories developed in Chapter 3, using the probabilistic models, lay the critical foundations for the rest of chapters. The main conclusions of Chapter 3 justify that the communication range of sensor nodes is the key element to control the topology of networks. Then, Chapter 4 presents the controllers to control the communication range, and meanwhile the performance of the proposals are evaluated through a Matlab-based simulator. However, the impacts of some parameters in the proposals on the performance of the whole system are unknown. Therefore, the following Chapter 5 tries to find heuristic algorithms to select appropriate parameters so that the performance is optimized. Chapter 6 implements and evaluates previous proposals in a real network that consists of Sun SPOT sensor nodes.

1.3 Contributions

This thesis addresses the resilience management problems in the IoT in terms of network connectivity in response to the disturbances. To make the IoT resilient against unexpected internal and external disturbances, there are many disciplines involved in this thesis. Several techniques in multiple subjects are leveraged, such as stochastic networks, control theory, optimization theory, and software engineering. The contributions of this thesis, therefore, cover distinct disciplines, and the main contributions of this work can be summarized as follows.

(1) Review the topology control theories and techniques that widely are used to study the fault-tolerance problems in large scale and ad-hoc networks, which would make this thesis a reference for researchers and practitioners who desire to search the state-of-the-art techniques to start the development of resilient network from the topology control perspective.

(2) Model the IoT as a stochastic network, because the deterministic model in a real network is very likely infeasible. This thesis presents a probabilistic model to describe the network connectivity, and provide novel insights through mathematical proofs.

(3) Experiment the fuzzy-logic, proportional-integral-derivative (PID) control, and self-adaptive software to design the self-adaptive systems in response to node failures. So, this thesis can serve as trials of some self-adaptive techniques to the IoT field.

(4) Test various heuristic optimization algorithms on the control system, which are proved to be effective. This may contribute to the system optimization for the IoT.

(5) The self-adaptive software design would be viewed as the complimentary to the control designs that extends the contributions of this thesis to the software field. It has been a nice opportunity to exercise the OSGi (Open Service Gateway Initiative) technique which possesses the reconfiguration functionality at runtime.

(6) The majority of the designs are experimented in the simulation environment (in particular, the optimization part), but fortunately, thanks to the European project "Design, Monitoring and Operation of Adaptive Networked Embedded Systems" (DEMANES) [41], part of the designs are tested in the real-world scenarios, which makes the work applicable to real-world applications.

(7) In order to disseminate the results, most of the contents in this thesis have been published or submitted at the time of writing this thesis to international journals and conferences, which can be found in "List of author's publications" at the end of the thesis.

(8) Lastly, the lessons learned from this research may be of great value. One can further consult the conclusions and future works sections.

1.4 Thesis structure

Before ending this chapter, the outline of this thesis is made as follows. The rest of this thesis consists of six chapters:

- Chapter 2 introduces the state-of-the-art related to this work, and discusses the models and the techniques which are employed in this thesis.
- Chapter 3 provides a mathematical model to analyze the stochastic network. Analytic research, together with numerical analysis in the case that the analytical approach is not feasible, are performed.
- Chapter 4 tries the fuzzy-logic controller to control the power transmission of sensors in order to create a fault-tolerant network, also the energy related problems are taken into account. The experimental results are demonstrated and discussed.
- Chapter 5 presents a lightweight controller to make the implementation more applicable. Besides, the controller is optimized by utilizing some bio-inspired algorithms.
- Chapter 6 shows the experimental results that are not simulation-based. The proposal in Chapter 5 is integrated with the self-adaptive software technique to form a self-adaptive middleware.
- Chapter 7 concludes this work, and points out the remaining challenges, which would be our future works.

Chapter 2

Related Works

This chapter provides a survey concerning the resilient design in the IoT from the topology control and routing protocol perspective. An assumption is implicitly made that nodes are wirelessly connected, which certainly is not true for all nodes in IoT. Nevertheless, this assumption is enough for many applications when studying the IoT. The latest review [155] classifies existing topology management techniques by reactive and proactive methods, but this chapter offers a different perspective and explores new solutions.

After completing the survey, Section 2.6 briefly clarifies which models are adopted in this research. This chapter itself is part of the contributions, because it provides a reference for those researchers who want to design the resilient IoT from both theoretical study and practical application point of view.

2.1 Network design flow

As a matter of fact, there are many ways to achieve resilience. For example, in the layered network model: physical, link, network, transport, and application layer, each layer can have its own resilient strategies. For instance, redundant coding enables detection and correction of coding errors; a multi-processor has a better fault-tolerance performance; the security mechanisms protect data confidentiality, integrity and availability, etc., but those methods are out of scope of this work. More specifically, this thesis reviews the state-of-the-art techniques on how to build a $k - connected$ network, where k is the most important indicator for resilience, and how to find out the alternative paths by using multi-path routing protocols. In general, resilience capability

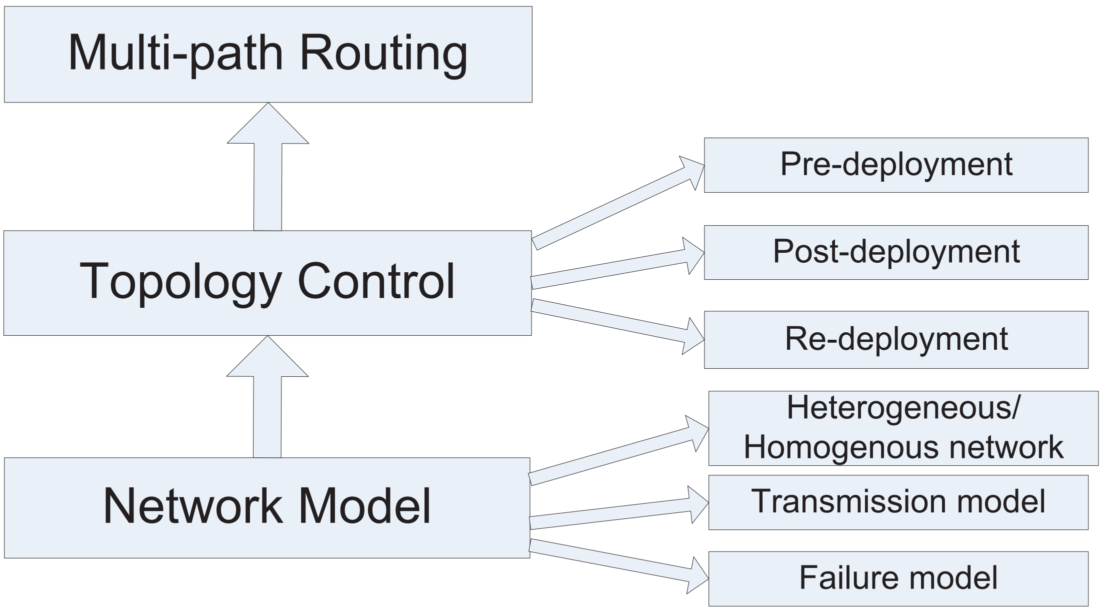


Figure 2.1: *Resilient IoT design flow.*

of the IoT relies on redundant deployment and adaptive design. Thanks to the low cost of the IoT nodes, unlike traditional networks, it is possible to deploy far more nodes than actually needed to make the network fault-tolerant. Researchers, implicitly or explicitly, leverage a network model; hence, first the network model in IoT is discussed. The thesis shows that it is complicated to completely model a real IoT. Afterwards, this thesis reviews the theoretical results and representative topology control approaches to guarantee IoT to be $k-connected$ in three different network deployment stages, as summarized in Figure 2.1.

(1) *pre-deployment stage*: The literature focuses on either how many nodes are needed if the nodes are randomly deployed or how to construct a particular network topology if nodes are able to control their positions;

(2) *post-deployment stage*: The main focus is to maintain or improve the node connectivity or link quality by adjusting transmission range or transmission power;

(3) *re-deployment stage*: It aims at how to populate new nodes to recover from the network disconnection.

Usually, network resilience is not the only concern, other problems such as network coverage, node energy, number of nodes deployed are also considered in the literature, which will be involved in the discussion when necessary.

Considering that multi-path routing protocols are important to resilient IoT and also one of the most active research areas in IoT, this part is involved in this survey. Note that the network topology is the foundation of routing protocols. If multi-path does not exist from the topology perspective, it will not be able to find alternative routing paths, no matter which multi-path routing protocol is used. Unfortunately, most problems related to the topology control have been proven to be very difficult to solve; some of them are either NP-completeness or NP-hard problems.

2.2 Network model

In this section, the general network model related to the IoT is introduced. The network models are quite important to analyze the IoT, because different models may lead to different conclusions.

2.2.1 Heterogeneous and homogenous network

In a *homogenous network*, all nodes have the same performance in terms of communication range, power level, processing ability, mobility, memory, etc. Otherwise, it is called *heterogeneous network*. Sometimes, in order to simplify the problem, one aspect is stressed and others are ignored. For example, if each node has the same range, the network is homogenous, otherwise it is heterogeneous. In practice, IoT-based deployments are very likely to be heterogeneous. For instance, the base station and cluster heads are more powerful than other nodes; hence, they have longer communication range than regular nodes; especially, the sink node has no constraints with respect to energy as it is usually not a battery powered device.

2.2.2 Transmission model

In most cases, researchers employ ideal transmission models for IoT, e.g. the nodes transmission model and sensing model are disks, the links between nodes are symmetric, and the transmission radio is omnidirectional; unfortunately, a real transmission model that produces the real world would be much more complicated.

◇ **Irregular radio model**

The transmission area of sensors is usually modeled as a disk centered at the node with its radius being its communication range, as shown in Figure 2.2(a). However, the reception signal power attenuation relies on $R^{-\alpha}$, where R is the transmission range and α is the loss constant that depends on the wireless medium which typical value is between 2 and 4, and can vary from device to device [125][56]. In fact, it has been found that the radio communication range is highly probabilistic and irregular [54][162], especially in the indoor environment [55], as shown in Figure 2.2(b). [162] and [1] show the evidence that the radio is irregular, even time-varying while stationary, in the MICA2 sensor platform. Most simulators do not have irregular radio models because of the complexity to analyze them. Our study [66] proposes a probabilistic approach to describe the connection between nodes, since the radio model is irregular. More specifically, the network connection probability is modeled as a Cox process, and the numeric simulations are performed to evaluate the network connection probability.

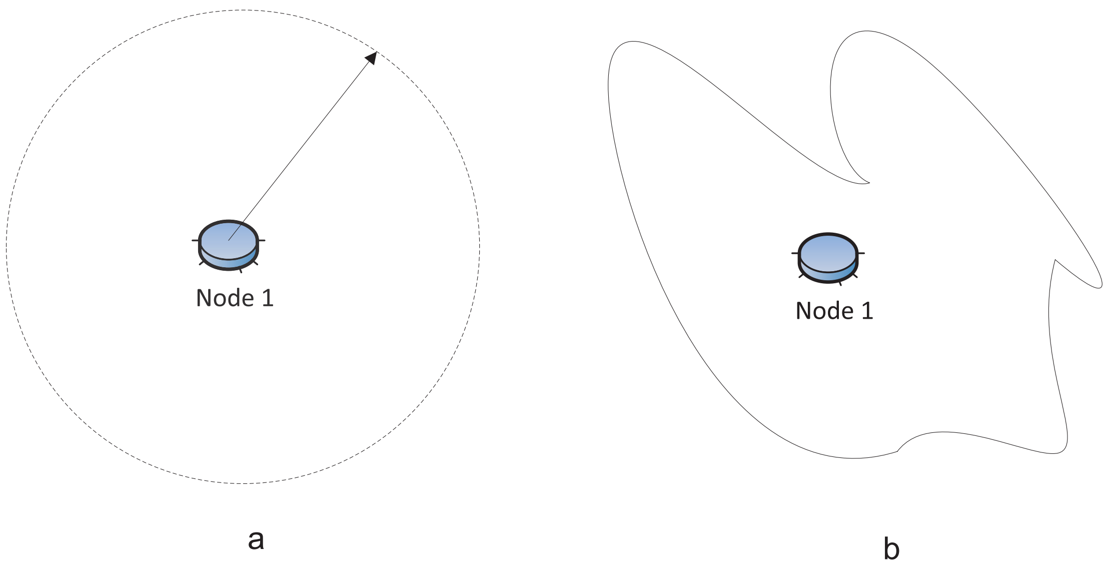


Figure 2.2: *Regular and irregular radio model.*

It is worth noticing that literature [125] states that the wireless connections can not simply be modeled as a geometric graph where each node connects other nodes within its communication range. For the wireless network, only the maximum outgoing edge contributes to energy, but other nodes with a shorter communication range can be connected more or less for free. This leads to the study of energy-related issues in the IoT that are quite different. The wired network has a link-based or edge-based metric, while the wireless network is node-based metric [125].

◇ **Asymmetric link**

The symmetric link implies that if one node is able to receive a message from another node, then it can send a message to the same node. For a wired network it is true, but for a wireless network it is true only when the communication ranges of both nodes are the same or the distance between two nodes is less than the minimum communication range of both nodes. The latter is called mutual inclusion graph (MG) in [90]. However, in any real case the network is likely to be asymmetric, so the resulting network is directed. The irregular communication range leads more frequently to an asymmetric link. Figure 2.3(a) and (b) show the symmetric and asymmetric communication models, respectively.

The omnidirectional radio indicates that the node's antenna broadcasts to all directions. In IoT, it is possible that radio communications can be modeled as a directional antenna which can transmit in specific direction(s), as illustrated in Figure 2.3 (c).

The wireless communication model is very important for the neighbor-based research. To obtain the analytical results, simplified assumption (e.g. disk-like and symmetric link) is allowed to get the approximation; while in case that a more realistic

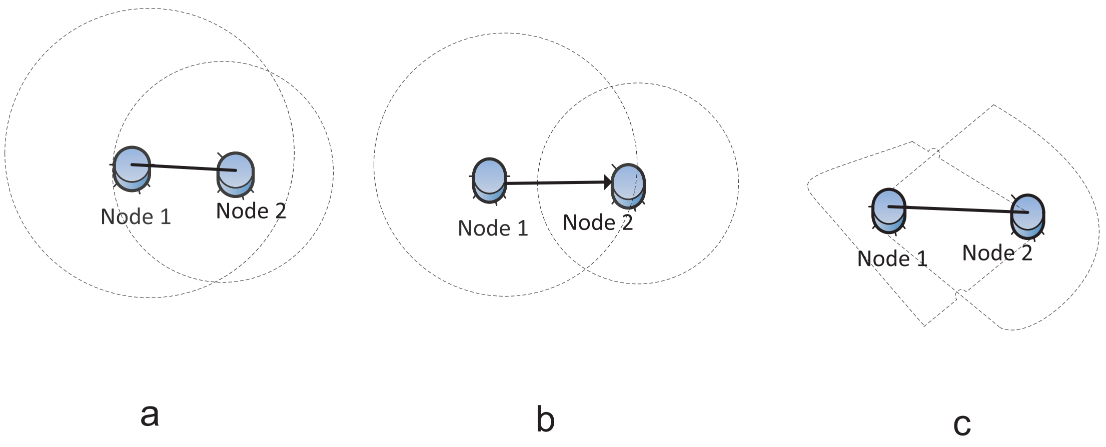


Figure 2.3: *(a) Symmetric link: Node 1 can communicate with node 2 and vice versa. (b) Asymmetric link: node 1 can send data to node 2, but node 2 cannot send data to node 1; (c) Non-omnidirectional radio.*

communication model is needed, numerical analysis often is the only mean to evaluate networks.

2.2.3 Failure models

Communications over wireless channels are more insecure and susceptible to various kinds of attacks compared to wired networks, e.g. the eavesdropping is easier on the wireless link. There are many reasons that cause network node(s) or link(s) failure. The failures may be caused by physical damage, such as fire, animals or vehicular accidents [23], or disasters like deluges, earthquakes, sandstorms, etc. In the network, the failure can be modeled as the removal of nodes or links, or as the probability of nodes (links) to fail. The difference is that when the node is removed, the corresponding links connected to it must be removed as well. [136] considers the statistically independent and dependent link failures under low stress (probability lower than 0.2) and high stress (probability more than 0.5) conditions, but without considering the node failure. The node failure model and link failure model seem quite different. However, [98] shows that the link failures are not qualitatively different from what is observed from the node point of view.

The failure model is very important and has significant influence on the algorithm design. One proposed algorithm possibly is only suitable for a specific failure, e.g. a single node failed at one time. In practice, the malfunctioning of a single component of a real system can generate a cascading effect. [49] considers two failure types: (1) isolated failures, where each node in the multi-path has a probability of failure during some small intervals T; and (2) patterned failures, with the failure of all nodes in a circle within a given small time interval T which is Poisson distributed. The isolated failure models the independent node failure; the patterned failure models the geographically correlated failure. [7][84][64] consider simultaneous failures. Nevertheless, a large number of nodes failing simultaneously is a low probability event [54].

The failures are classified into permanent failures, transient failures, and faulty readings. To detect the node or link failure, the most common approach is to broadcast the messages and wait for the response in a certain time interval in order to test the connectivity to other nodes. For instance, in the routing protocol Octopus [113], the packet is broadcasted by each node every hello-timeout period. If a node does not hear from a neighbor longer than two times hello-timeout, it removes this neighbor from its neighbor list. Nevertheless, the failure node detection is not the focus of this thesis. A survey on failure detection is referred to [101].

◇ **Permanent failure model**

In the permanent failure model, nodes fail forever and have no chance to recover. The permanent failure model can be further categorized into: (1) random failures, where the failure occurs randomly in the IoT; and (2) attacks, where the failures are due to deliberate damage, such as preferentially targeting most connected nodes. Random damage can be modeled as randomly removed nodes or links, but attacks are modeled by removing nodes or links following some strategies, for instance, remove the nodes which have higher degree [98][160]. [116] categorizes the attacks based on their impacts, including data integrity and confidentiality, power consumption, routing, identity, privacy and service availability.

[104] analyzes that the single failure probability dominates the overall disconnection probability in a regular graph where each node has the same degree. In a random network with the size tending to infinity and the degree distribution being p_k, after the removal of a fraction p of the nodes during a classical attack (such as remove the nodes with the highest degree), the maximal degree $K(p)$ satisfies (2.1) [98]:

$$p = 1 - \sum_{k=0}^{K(p)} p_k \tag{2.1}$$

◇ **Transient failures model**

In the transient failures model, the failure occurs but the system reverts to its former status after a time interval, or fluctuates between a normal and an abnormal status. [113] takes into account when the network is intermittently disconnected and connected. Each time an unstable node awakens, it remains connected for a time interval chosen uniformly at random in the range [0, 120]m. When it is disconnected, it remains disconnected for a time interval chosen uniformly at random in the range [0, 60]m.

◇ **Faulty reading**

The nodes operate well but perhaps are suffering from faulty readings. The sensors with faulty readings are called faulty sensors. Their malfunctioning can be attributed to: (1) faulty data measurement or data collection; (2) some variables in the area surrounding the sensor have changed significantly; (3) inherent performance of the sensor is abnormal [42]. The faulty data can only be referred to as deviation from the expected model [106]. More type definitions of the faulty data can be found in [106]. To detect faulty nodes, the stochastic method is usually employed. For instance, given a defined threshold, its value is compared to the values of its neighbors and the median value is used to filter out the faulty measurement, e.g. [96][42][43][83].

2.3 Topology control

From the network topology point of view, the resilient IoT problem is a $k-connected$ problem. The topology control is a common way to preserve or improve connectivity of the IoT by controlling network topology. Note that the connectivity is one of the most important, but usually not only concern in the IoT design. Some other parameters, such as coverage, energy and number of nodes required, also should be taken into account in the optimality process. In most cases, both of the network coverage and connectivity have to be considered, e.g. [134][54][62][34]. One of latest surveys about coverage problem in IoT can be found in [133].

Generally, the connectivity is more important than other performances because the data collection is always the first priority for IoT. The coverage and energy problems become less important if the sensor nodes are not able to forward the collected data to the destination.

2.3.1 Topology control in different deployment stages

There are three deployment stages for the IoT: pre-deployment, post-deployment, and re-deployment. For each stage, there are corresponding approaches to preserve, improve, or recover network connectivity:

(1) *pre-deployment stage*, that is, the sensor node positions are not fixed yet, and the network is not operational, e.g. it has not started sensing the environment. At this stage, one can deploy sensor nodes randomly or construct a particular network topology to satisfy the required connectivity.

(2) *post-deployment stage*, that is, the sensor node positions are already known. At this stage, one can adjust the transmitting range, or equally, node transmitting power to achieve desired connectivity. The goal usually is to improve the connectivity while consuming as little energy as possible or improve the link reliability, as shown in Figure 2.4.

(3) *re-deployment stage*, that is, new nodes can be added in the original network. At this stage, one can populate new nodes in the original network to maintain or repair the network connectivity, as shown in Figure 2.5. Usually, the goal is to improve the

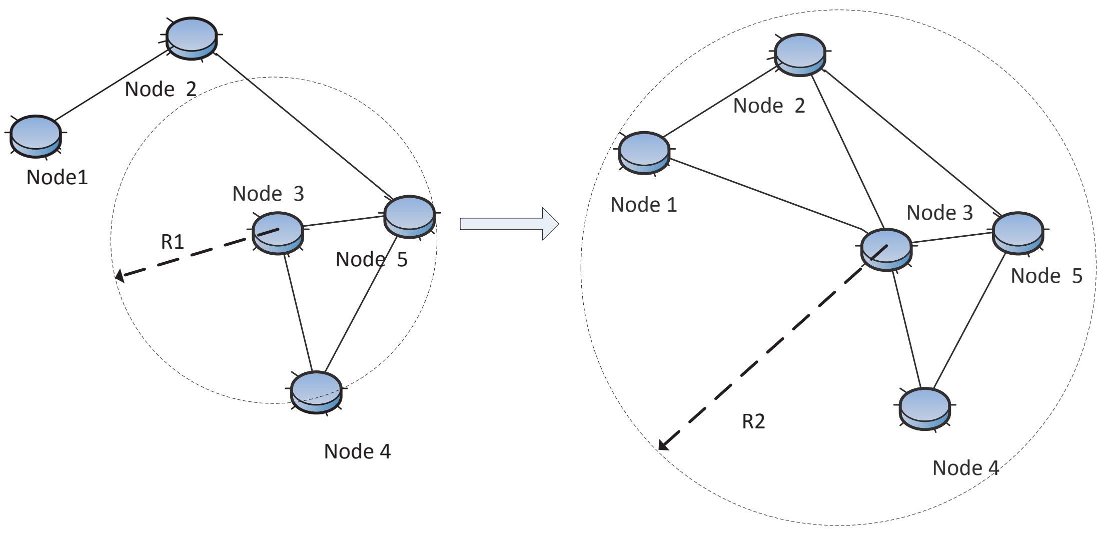


Figure 2.4: *Topology control: adjust the transmission range of node 3 from R1 to R2 in order to improve connectivity.*

connectivity while using the number of nodes as low as possible.

Commonly, the connectivity improvement strategies will be applied onto only one stage. Unfortunately, all of them are challenging works because most of them are either NP-completeness or NP-hard. In Section 2.1, this thesis lists many problems regarding the network connectivity.

2.3.2 Pre-deployment

Usually, there are two cases in the pre-deployment stage: nodes are randomly deployed or their locations can be controlled.

◇ **Random deployment**

If the sensor nodes are randomly deployed, it is usually assumed that the node position has uniform distribution. The question is how many neighbors (or node degree) a node needs so that the whole network will be $k-connected$. In the uniform distribution network, [105] states that 6 to 10 neighbors are able to make sure that the network is connected with high probability. [144] concludes that the "lower bound" to make network disconnected and "upper bound" to make network connected are $0.074 \log n$ and $5.1774 \log n$, respectively, where n is the total number of nodes. [27] observes the upper bound is $\alpha e \log n$, where $\alpha > 1$, e is the nature base. [24] improves in the sense that if $k \leq 0.3043 \log n$, then the graph is not connected with high probability and if $k \geq 0.5193 \log n$, the network is connected with high probability as $n \to +\infty$. The $k - connected$ network is asymptotically the same as that for $1 - connected$ [16], which indicates that once the network is $1 - connected$ it achieves $k - connected$ immediately in the uniform distributed network. However, those asymptotical conclusions hold

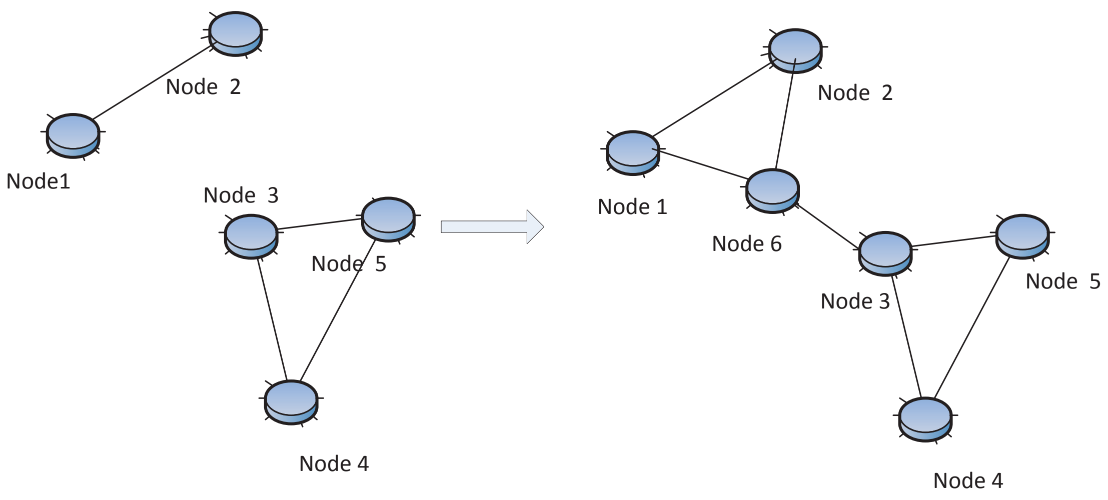


Figure 2.5: *Topology control: new node 6 is added to recover connection.*

theoretically only when $n \to +\infty$, which is not so practical.

Our study [66] (the details will be discussed in Chapter 3) leverages a probabilistic approach to analyze the network connection probability, and proves that if the probability of the network being connected is 0.36ε, then the probability of the network being connected is at least $1-\varepsilon$ by means of increasing communication range by constant $C(\varepsilon)$, where $0 < \varepsilon < e^{-1}$. Explicit function $C(\varepsilon)$ is found. Furthermore, the localized control algorithms based on fuzzy-logic are proposed in our papers [65] and [64] to achieve desired node degree. The simulation results show that the average node degree can be achieve in dynamic IoT due to the close-loop feedback of the control system. The proposal in [63] improves the fault-tolerate performance in the presence of random attacks. By combining the theory in [66] with the control algorithms in [65][64], it is possible to design a network being $k-connected$ with high probability, which can tolerate at most $k-1$ node failures.

If the nodes are uniformly and randomly deployed, the degree distribution follows the Poisson distribution, but in real life degree distributions are strongly non-Poisson distribution, often taking power law, truncated power-law, or exponential forms [28]. [36] shows that the degree of nodes in technological networks, such as the World Wide Web, the Internet, and airplanes connection networks is power-law distribution. Some biological systems, such as metabolic networks and protein networks, are different from the uniform distribution networks and have a power-law degree distribution with an exponent that ranges between 2 and 3. The power-law degree distribution is more fragile to deliberately attack than the Poisson degree distribution, but the power-law degree distribution is more resilient to the random failure than Poisson degree distribution [28][6][36]. The reason is straightforward. For the uniform distribution networks, each node contributes equally to the network, so randomly removing nodes cause the same amount of damage. By contrast, for the non-uniform distribution network, each node plays a different role, and attacks the most connected nodes (namely higher degree) causing heavy damage. [33] shows a real case of a Japan earthquake and tsunami. In this case, Internet was impressively resilient to the disaster due to its heterogeneity. However, [98] finds that when the number of nodes is a limited value, the difference between Poisson and power-law networks in practice is not that significant; instead, it is overestimated. The network is more vulnerable if the network scale increases but the degree of a regular network is constant, which implies that a large-scale network is more susceptible than a small-scale network if they both have the same degree. It is believed that, like other networks, IoT also are resilient to the random failures but fragile to attacks.

◇ **Construct $k-connected$ network**

When the nodes in the IoT are able to control the locations, the questions are how to construct a $k-connected$ network or how to verify that the constructed network is $k-connected$. At the same time, it is necessary to consider the $k-coverage$ issue in some cases.

(1) *Construct $k-connected$ and $k-coverage$ network:*

There is a well-known result showing that if the communication range is at least two times longer than the sensing range, that is $s_c \geq 2s_s$, then $k-coverage$ can guarantee $k-connected$ [134]. If the nodes location is carefully designed, the ratio can be further optimized. For example, [12] proves that strip-based is an asymptotically optimal deployment for achieving both $1-coverage$ and $1-connected$ for all $\frac{r_s}{r_c} \leq \sqrt{3}$, and [141] proposes Diamond pattern can achieve the asymptotically optimal for $1-coverage$ and $4-connected$ when $\frac{r_s}{r_c} > \sqrt{2}$. For higher connectivity, [13] presents the hexagon-based deployment pattern to achieve $1 \leq k \leq 6$ connectivity. But the optimal ratio between communication range and sensing range such that the network is $k-coverage$ and $k-connected$ for general k, is still an open issue.

[62] first shows the conditions that network is $k-coverage$ and $k-connected$. It claims that the network is $k-coverage$ and $k-connected$ if it is $k-DPC$. $k-DPC$ is defined as: if each point on the perimeter of every node is covered by at least k other nodes through sensing range, meanwhile each node has a link to its all neighbors. Here, two nodes being neighbors means that there exists an intersection between their sensor ranges, rather than the communication ranges. It proposes the distributed protocols to guarantee $k-coverage$ and $k-connected$ under the condition that the initial coverage and connectivity is higher than expected. Similarly, [71] also proves that the target field is $k-coverage$ only when the sensing border of every node is $k-coverage$. It uses this conclusion to verify whether the network is $k-coverage$, or a coverage hole exists in $d-dimension$ network, where d is 1, 2, 3.

On the contrary, known the ratio between r_c and r_s, the connectivity can be calculated. [54] suggests that if communication range r_c is at least equal to sensing range r_s and the network is connected, its connectivity k is

$$\frac{2k\pi r_c^2}{(\pi-\sqrt{3})r_s^2}$$

for homogeneous $k-coverage$ network,

$$\frac{2k\pi R_{max}^2}{(\pi-\sqrt{3})r_{min}^2}$$

for heterogeneous $k-coverage$ networks, where r_{min} and R_{max} are minimal sensing range and maximal communication range respectively in a heterogeneous network. The sufficient density to guarantee $k-coverage$ for a homogeneous network is

$$\frac{2k}{(\pi-\sqrt{3})r_s^2}$$

and

$$\frac{2k}{(\pi-\sqrt{3})r_{min}^2}$$

for a heterogeneous network [54].

(2) *Construct new $k-connected$ network from existing $k-connected$ network:*

Let G be a $k-connected$ graph, if H is a graph obtained from adding a new vertex and joining it to at least k vertices of G, H is also $k-connected$ [3]. In Figure 2.6, the original G is $2-connected$ network and the new graph H is constructed by adding a new node 4. There are two disjoint links connected to G from node 4, so H is a $2-connected$ network. Likewise, two disjoint $k-connected$ components form a large $k-connected$ component if there are $k-vertex$ disjoint edges connecting them. There are several papers leveraging this theorem. This theory can also be used at the post-deployment stage; it will be discussed in Section 4.3. [23] presents a distributed algorithm by locally constructing $k-connected$ components and then joining two components by k disjoint edges between them to form one larger $k-connected$ component. [25] proposes an algorithm to construct new network topology by joining two or more constituent route graph. For instance, starting from really simple networks having 2, 3 and 5 nodes, it is able to construct a network that has cardinality N in the form $2^i3^j5^k$; the resulting graph can tolerate $i+2j+2k-1$ faults.

(3) *Spanning tree:*

The spanning tree is a tree that connects all vertexes in the graph. One graph may have more than one spanning tree, while the one has minimal total weight called Minimal Spanning Tree (MST). Since it is a tree, it is a $1-connected$ network. Therefore, if one only emphasizes on the $1-connected$ and minimal cost (e.g. energy), it is modeled as MST problem, which can be solved by using well-known Kruskal's and Prim's algorithm

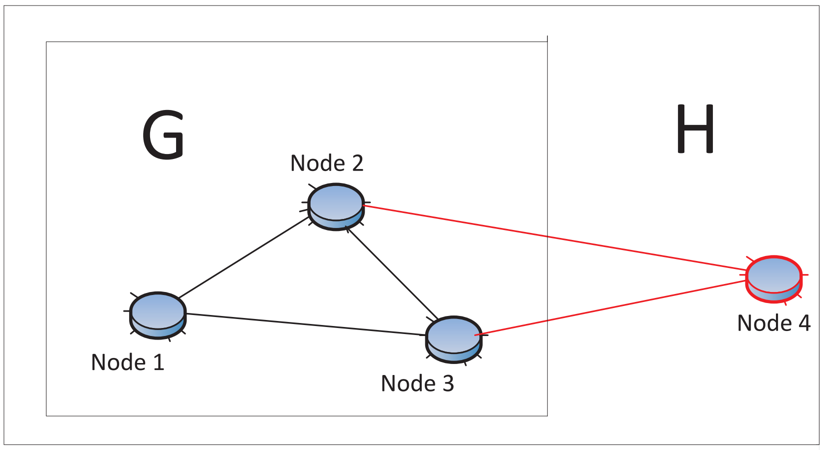


Figure 2.6: *Construct new* $2-connected$ *network* H *from existing* $2-connected$ *network* G*.*

[143] and a distributed algorithm in [77]. MST can be solved with $O(|V|)$ by linear time randomized algorithm [70]. But for $k = 2$, the problem finding minimal $k - connected$ spanning sub-graph is NP-hard and has $1+\varepsilon-approximation$ solution [78]. There is no polynomial time approximation less than $1+\varepsilon$ for any given positive constant ε. Let the number d_{MST} be defined as the maximum possible degree of a minimum-degree MST spanning points from the space (e.g. if in Euclidian plane, d_{MST} is 5), the algorithm for $2 - connected$ and $2 - edge - connected$ is $2d_{MST} - approximation$ [72].

2.3.3 Post-deployment

After the nodes are deployed, the topology control can be achieved by adjusting transmission range (power) or scheduling the nodes status to be active or inactive. This ability is achievable in most sensor platforms, such as Crossbow and Sun SPOT. However, increasing the transmission range also causes higher signal inference. Consequently, it has negative influence on the link reliability.

⋄ Connectivity, energy-efficiency and coverage

In addition to connectivity, there are several other issues that should be taken into account, such as the lifetime of nodes. Higher transmission power results in faster energy consumption. For example, network connectivity is the largest when each node transmits at its maximum power, but it has the shortest lifetime. Extending the lifetime of the network in some scenarios is a desirable requirement since the sensor nodes may be deployed in areas where human are not allowed to enter in, or the nodes cannot be replaced, or the battery cannot be recharged. Besides, if there are many redundant nodes, the coverage problem arises when some nodes are scheduled to be active, and others be inactive, which is a common approach to maintain the network connectivity and coverage.

(1) *Adjust transmission range:*

The purpose is to assign to each node a transmission power so that the resultant network is $k - connected$ and meanwhile the transmission power is optimized. Optimizing energy while keeping network connectivity strong is a challenging task [49][125][103][122][91]. Most of the energy optimization and $k-connected$ problems have been proven to be NP-completeness [125]. Note that minimizing the total transmission power assignment is NP-hard even for $1 - connected$ network [32][123]. There are centralized or distributed, or hybrid (e.g. [32]) approaches to solve this problem.

If the node transmits data with maximum transmission power and finds out the k optimal vertex disjoint path to each neighbor is according to certain criteria (e.g. power cost), [115] proves that if each node maintains k optimal disjoint paths, the resultant network are $k-connected$ globally, provided that all nodes use the maximum transmission power. In [32], the $k-connected$ network are formed by three steps: first, cluster the network; second, the cluster head assigns each node a communication power such that each node in the cluster is $k-connected$, it is called the intra-cluster topology control; third, it is ensured that there are $k-disjoint$ paths between adjacent clusters, this is called inter-cluster topology control. After three steps, the resultant network is $k - connected$ globally. [128] aims at building up and maintaining $k - regular$ and $k-connected$ nodes by adding links between overlay node. This approach forms groups consisting of k nodes, and each group is a $k - regular$ network, it regards the group as single node, then recursively using the same approach to form the final topology.

If the network already is a $k - connected$ network, what is the condition preserving $k - connected$ by changing power based on the angle of neighbors? [21] studies the angle of its neighbors. The cone of degree of nodes determines the connectivity. A node u transmits with the minimum power P, in order to ensure that in every cone of degree around u there are some nodes that u can reach with power P. It is shown that taking angle $\frac{5\pi}{6}$ is a necessary and sufficient condition to guarantee that network connectivity is preserved. [21] tries to use fewer edges by removing edges but preserve connectivity. Two basic mechanisms are adopted [21]: (1) start from initial power, then increase power until connected; (2) start from the largest energy then reduce it as long as connectivity is not changed. [103] preserves the connectivity of the network and meanwhile minimizes transmission power. [103] proves that the upper bound of angle is $\alpha \leq \frac{2\pi}{3k}$. Note that many algorithms try to optimize energy inevitability incurring an extra overhead.

(2) *Schedule the duty cycle of nodes:*

Another common method to optimize energy is to schedule the duty cycle of nodes. There are usually three status in each sensor node: active, idle, and sleep. The node in sleep or idle mode consumes less energy than the active model. Nodes are scheduled to keep some nodes alive or active if they are on duty, while letting other nodes in sleep mode in order to save energy of the whole network, but still keeping the network connected. [29] points out that the idle status consumes almost the same power as the active status, but the sleep mode consumes much less than active status.

If every node is active with equal probability p, the asymptotic $k-coverage$ results for grid, random uniform, and Poisson deployment are studied in [79]. [34] utilizes random scheduling for required coverage, but the active nodes may not be connected, therefore other nodes are turned on to guarantee connectivity if necessary. The node A is called the upstream node of B if A is neighbor of B and one hop closer to sink; accordingly, B is called the downstream node of A. The status of the node A is not only decided by the random scheduling assigned by the coverage-scheduling, but is also determined by its downstream nodes. For example, node A may be in a sleep model assigned by coverage-scheduling, but it will be forced to switch to active mode if its downstream node is active and there is no path for this downstream node for transmitting data to the sink.

The model in game theory can be adapted to optimize energy of the IoT as well. For instance, the player in IoT is the node. All nodes work in non-cooperative manner. They selfishly conserve energy by refusing participants as relays nodes, but they can participant as a relay by offering some incentive to encourage cooperative. An incentive could be the token. The tax mechanism can be adopted as well: higher communication range introduces higher interferences to neighbor's nodes, hence need to pay high tax. [97] surveys game approaches to formulate problems related to security and energy efficiency in IoT.

◇ **Link quality**

Increasing the network connectivity by power control is one way to improve the IoT resilience, but the individual link quality is also important for resilient IoT. Many power control approaches are employed to improve the link quality, thereby improving the reliability of the link. Higher transmission power helps to improve data transmission reliability, but also introduces higher energy consumption and interference. There is tradeoff among energy efficiency, link quality, and interference.

There are papers dedicated to conduct practical experiments aimed to improve link quality, such as [55][153][93][139]. ART [55], an adaptive topology control protocol, is based on empirical study of the link quality in the indoor environment. It adapts the transmission power in response to variations in the link quality triggered by either environmental changes or varying degrees of network contention. ART monitors the outgoing packets and records the transmission failures during a defined time window. If the transmission failure is out of the defined threshold, ART increases or decreases the transmission power. One of the advantages is that ART does not always increase the transmission power to improve the link quality because, in practice, it is not always true under high contention and interference. Instead, it may decrease the transmission power under high contention. [153] proposes a protocol that applies a control-theoretic approach to control packet reception ratio (PRR) directly. The receiver monitors all incoming packets, records the packet transmission failures and then computes the transmission power needed next round. This information will be transferred to the sender side. [139] adjusts the transmission power by measuring the average received signal strength indicator (RSSI), and comparing it to upper and lower thresholds to decide nodes increase or decrease the transmission power. Adaptive Transmission Power Control (ATPC) [93], employs a feedback-based transmission power control algorithm to dynamically maintain individual link quality over time. It proposes a protocol to minimize the power level by choosing proper transmission power for each packet transmitted, while guarantees the desired link quality. First, [93] reveals the relation between the transmission power and link quality indicator RSSI and Link Quality Indicator (LQI), which is generally monotonic and continuous but not deterministic. There are fluctuations in a small range at any transmission level. Based on the empirical results, [93] formulates a predictive model to characterize the relation between transmission power and link quality. With this predictive model, the sender node adjusts the transmission power in order to guarantee a specific receiver the desired link quality. [142] provides a localized Configurable Topology Control (CTC) algorithm to achieve desired path quality bounds in lossy IoT.

⋄ **Connectivity maintenance protocols**

There are many energy-aware routing protocols. In order to balance the connectivity and lifetime of the network, a good energy-aware protocol should: first, schedule as many nodes as possible to turn off; second, keep the network connected; third, offer as much total capacity as the original network; fourth, power saving should inter-operate correctly with whatever routing system [29]; fifth, the algorithm needed to be distributed and highly efficient.

In SPAN [29], a node volunteers to be a coordinator if it discovers that two of its neighbors cannot communicate with each other directly or through an existing coordinator. CCP [134] protocol is a decentralized protocol that only depends on local states of the sensing neighbors. CCP is able to guarantee both coverage and connectivity if the sensing range is less than half of the communication range, but if the sensing range is more than half of the communication range, CPP integrates with SPAN [29] to provide both coverage and connectivity. PEAS [151] schedules a required number of nodes awake by adjusting sleep period in the presence of failures. It sends PROBE messages to the nodes within probing range which is a value lower than the communication range. Any nodes within its probing range will send back a REPLY messages. The sleeping nodes keep working only if they do not receive the REPLY messages. In this way, it keeps the network connected. [23] presents a greedy algorithm: it starts from the complete graph, then reduce nodes to reach desirable connectivity, or start from the empty graph and later connect edges until desirable connectivity is reached; then it cuts nodes but maintains the same connectivity. The second step is to reduce the redundant nodes.

2.3.4 Re-deployment

In last sections, it is assumed that deployed IoT nodes can not get in touch or relocate or add nodes once they are deployed. IoT can adapt to keep connectivity. However, for real IoT nodes deployment, it is still possible that nodes are able to be added or relocated in some circumstances, either by human beings or robot-like devices carrying new nodes to the field. The goal of the node placement is to achieve some desired performances, such as energy optimization, maintaining or improving connectivity by lowest number of nodes. Adding new nodes or edges set to make graph being $k - connected$ is also called connectivity augmentation problem. The boundaries are given in [51]: $\frac{2n}{3}$ additional edges are required in some cases and $\frac{6n}{7}$ additional edges are always sufficient, where n is the total number of sensor nodes. The connectivity augmentations can be done by populating new nodes to achieve a particular architecture which has strong connectivity. For instance, the goal of a new node may be to construct a complete network where each node connects to all other nodes, or regular graph which each node has the same number of neighbors, or circulates like Harary graphs.

[154] categorizes the placement strategies into static and dynamic depending on whether the optimization is performed at the time of deployment or while the network is operational, respectively. The behavior of mobile nodes can be the synchronous model, or asynchronous model, or semi-synchronous [40]. In the synchronous model, all relay nodes take action at the same time; in the asynchronous model, all relay nodes act individually; in the semi-synchronous model, a subset of nodes moves but not all.

◇ **Integer linear programming**

The node placement problems can be formalized to become a linear programming (LP) problem or integer linear programming problem (ILP). ILP could be a optimal approach to find the optimal numbers or positions of new nodes to improve or maintain connectivity if the network is small, but for a large network the LP or ILP are not applicable.

For instance, the energy optimization for $k - connected$ network can be regard as a LP problem. The constraints and objective functions are given in [56]. [15] formulates the optimal number of relay nodes to guarantee connectivity and coverage to be an ILP problem. [52] investigates the node placement strategies to achieve five objectives: mean overall residual energy, mean minimum residual energy, mean number of nodes over the threshold, mean overall travelled distance, and mean maximum distance. The problem is formalized to be a ILP, but unfortunately, it has huge numbers of constrains, including positions of all deployed nodes, and their status information.

[18] proposes a ILP method to solve minimal numbers of relay node placement

issues related with energy consideration to guarantee network connectivity. There are 20 constraints and many variables depending on the number of the relay node positions. It shows that for k is small (1 or 2) and the network is not that big (tens of nodes), the problem is solvable within reasonable time. However, if the network is large, and for bigger k value, it seems that it is still intractable by ILP. Therefore, few problems can be solved in polynomial time. In turn, most of them are very difficult to solve with the optimal solution. If the original problem is NP-completeness, the corresponding LP or ILP is also NP-completeness.

◇ **Steiner tree**

Given a weighted undirected graph and a subset of terminal nodes, finding a minimum-cost tree spanning terminals, is called Steiner tree problem, which is NP-hard. The difference between Steiner tree problem and the MST problem is that Steiner tree allows to construct or select intermediate connection points to reduce the cost of the tree. Those intermediate nodes are the optimal new nodes that the network needs to maintain the connectivity and low cost. For now, the best algorithm has $1.39-\mathit{approximation}$ for weighted Steiner Tree [26]. MST can be used to solve Steiner tree but is not necessary to be optimal solutions. Placing minimum number of nodes for maintaining network connectivity such that the transmission range of each sensor does not exceed a threshold is Minimum number of Steiner Points (SMT-MSP) [31], which is NP-Hard as well. Many nodes placement optimization problems can be reduced to SMT-MSP problem, e.g. [118][7][84][40][92].

◇ **Heuristic algorithms**

Node optimal placement is a very challenging problem. [40] claims that no local algorithm can achieve good approximation factors for the number of relays. Many of them have been proved to be NP-completeness or NP-hard, even only want to achieve $2-\mathit{connected}$ network [20][2].

Since most relay nodes optimizations are NP-completeness or NP-hard problems, many heuristic solutions have been proposed. [92] first uses a minimal spanning tree algorithm over points set P, then inserts new points for those edges longer than R, resultant tree has no edges longer than R. This algorithm has $5-\mathit{approximation}$. [31] improves the approximation of [92], and proposes two algorithms which have performance approximation 3 and 2.5 respectively. Note that they simply consider the $1-\mathit{connected}$ since it is a tree. Similarly, in the heterogeneous networks the nodes with distinct transmission ranges, the node placement for full $k-\mathit{connected}$ (consider all nodes) and partial $k-\mathit{connected}$ (not consider the populated nodes) are NP-hard problems [140]. The algorithms proposed in [140] have two phases: 1. solve the minimum $k-\mathit{connected}$ spanning graph for initial network; 2. put necessary relay nodes calculated graph in the first phase. According to first phase, it's not difficult to know where and how many relay nodes need. However, those methods need to know the location of each node.

[20] proposes $O(D \log n)-approximation$ replacement algorithm of minimal number of relay nodes for $2-connected$ network, where D is $D(1,2)-diameter$. Notice that it assumes that regular nodes have the same communication range, but the relay nodes have double range than other nodes. [40] tries to maintain the network connected while minimizing the number of relay nodes. It considers the global and local algorithms. Notice that there are two kinds of node: terminal and relay node. The difference is that the terminal nodes are stationary and range is 1, while relay nodes can move and have range more than 1. [38] proposes a relay nodes placement algorithm for $k-connected$ (not only $2-connected$) network based on the Particle Swarm Algorithm, also considering the heterogeneous feature of the IoT which nodes have different communication range. Notice that the algorithm runs off-line. [38] considers that heterogeneous IoT is only for original nodes, for all new joined relay nodes have the same communication range. [68] proves the algorithms have complexity $O(n^6)$ or $O(cn^3)$ for any k. [119] proposes two distributed relay node positioning approaches, based on virtual force-based movements and game theory respectively, to guarantee network recovery by minimizing the movement cost of the relay nodes. [107] aims to position the minimum number of additional relay nodes by the branch-and-cut based algorithm, such that signals from sensor nodes communicate to the base station within a pre-specified delay bound.

2.4 Multi-path routing protocol

Once the sensor nodes are deployed in the area of interest using the approach in Section 2.3, the network is $k-connected$ from the topology point of view. The routing protocols are used to find a specific path from the source to the destination according to specific routing algorithms. When the nodes or links fail on this path for whatever reason, the routing protocols need to detect failures first and then find alternative paths to the destination. As a consequence, the routing protocols play an important role in resilient IoT.

[8] clarifies that the routing protocols in wireless ad hoc and mesh network into different categories: proactive, reactive, geographical, multi-path, hierarchical, flow-oriented, WMN and Multicast, power-aware, etc. Of course, some of them share common properties. However, surveying routing protocols in IoT are out of the scope of this thesis, reader is referred to [8] for more details. Multi-path routing and energy-aware protocols, such as [46], REER [147], RELAX [148], LIEMRO [112], EEFTM [74], PEAS [151], are suitable for resilient IoT.

The challenges here are how to design low overhead and low complexity routing algorithms, taking into account heterogeneous or homogenous network models,

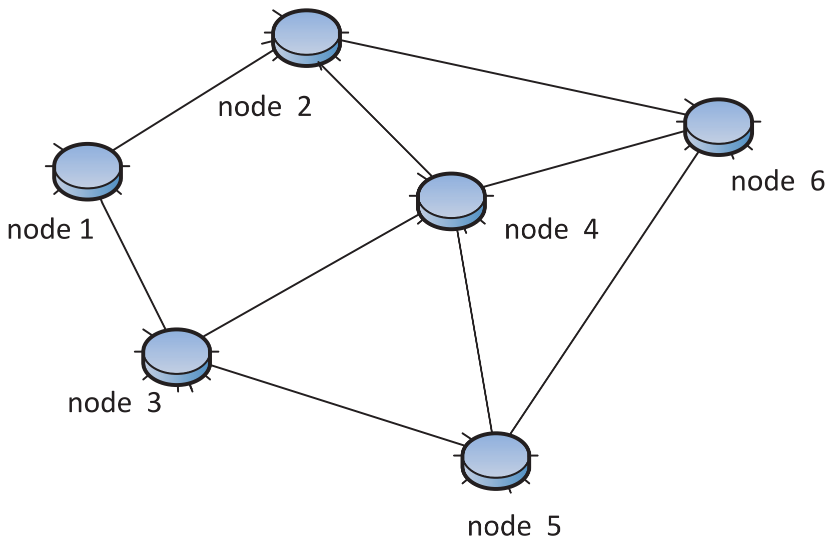


Figure 2.7: *Disjoint paths and brained paths. (1, 2, 6) and (1, 3, 5, 6) are disjoint-paths. (1, 3, 4, 6) and (1, 3, 5, 6) are brained paths.*

stationary or mobile networks, using 2D or 3D models, knowing the locations of node (e.g. by GPS) or not, localized (distributed) or centralized (globalized) algorithms, etc.

2.4.1 Complete disjointed and brained path

There are two popular routing protocols, DSR [114] and AODV [69] in IoT. DSR keeps multiple paths in the routing table for increasing reliability. This gives DSR good fault-tolerance. AODV uses multi-round discovery, exploring alternative paths to establish a route. However, both DSR and AODV are on-demand routing protocols, which may lead to significant latency and overhead (energy) at every routing path finding stage. Multi-path routing eliminates the routing discovery process unless there is no path available.

In general, there are two kinds of multi-path routing, as illustrated in Figure 2.7: (1) complete disjointed path: all paths do not share common nodes except the source and destination nodes. There are many researches on this topic. The algorithm to find $k-disjoint$ paths earlier can be found in [127][124][86][85]. (2) brained paths: the paths are not completely disjoined, there are many partially disjoint alternate paths. There is a tradeoff between resilience and maintains overhead [113]. For comparable resilience to patterned failures, the brained multi-path expends only 33% energy of complete disjoint paths, and has 50% higher resilience to isolated failures [49].

Formally, there are two ways to evaluate the relation between two paths: disjointness and stretch [85]:

$$disjointess = 1 - \frac{|P_{primary} \cap P_{backup}|}{|P_{primary}|} \tag{2.2}$$

$$strech = |\frac{P_{backup}}{P_{primary}}| \tag{2.3}$$

$P_{primary}$ and P_{backup} are defined as the set of links in the primary and the backup path. The resilient network needs *disjointness* close to 1 and stretch as small as possible. Paths must be chosen to achieve the highest reliability possible, though. [120] proves that in order to improve the reliability of multi-path routing, if the primary path is the shortest path, then the second path should be as short as possible; while if the primary path is not the shortest path, then the second path should differ than the first as large as possible.

2.4.2 Multi-path searching algorithm

If it is only required to confirm that the network is $1-connected$ or not, one can simply run the Depth-First Search, or Breadth-First Search, both have $O(|V|+|E|)$. [94] employs the Breadth-First Search algorithm to judge whether a node caused the network partitions, then the link rebuilding strategy starts to achieve the topology maintaining. [47] proposes an optimal algorithm to find the single-source shortest non-negative path with complexity $O(|E|+|V|\log|V|)$. Finding the maximum number of $k-disjoint$ path problem can be solved by network flow theory, specifically the minimum cut or maximum flow theory. There are many polynomial time algorithms solving maximum flow problems. So far, the best algorithm has complexity $O(|V||E|\log\frac{|V|^2}{|E|})$ [51]. The multi-path from source and destination for all nodes can be found in $O(kN^2)$ running time algorithm [125].

But for the IoT, there are always some special considerations, such as the computational capacity needed to think about before implement the algorithm, so $O(kN^2)$ running time perhaps is unacceptable in the IoT. Hence, the problems that have been solved in graph theory may be either hard or even impossible to be applied to the IoT due to having an unacceptable algorithm complexity. Therefore, the heuristic algorithms are more feasible. For example, a greedy algorithm can be: first time, the algorithm runs to choose one path to destination by selecting the best hop at each step, then running the same algorithm again to find the second path except that the node (link) has already been chosen, and so on, such as routing protocols [49], [149] and [146].

[49] realizes the multi-path routing by only using the local information, as shown in Figure 2.8. The source floods data to the sink, and then the sink knows which neighbors is preferred according to metric like lowest delay or loss, as shown in Figure 2.8 (a). Afterwards, the sink sends "primary path reinforcement" data back to the source. At this step, the neighbors choose their own preferred neighbor to forward data to the source. In the end, the primary path is established, as shown in Figure 2.8 (b) path P. After that, the sink sends the "alternative path reinforcement" to the source by choosing the second preferred neighbor, and along the way back to the source, each

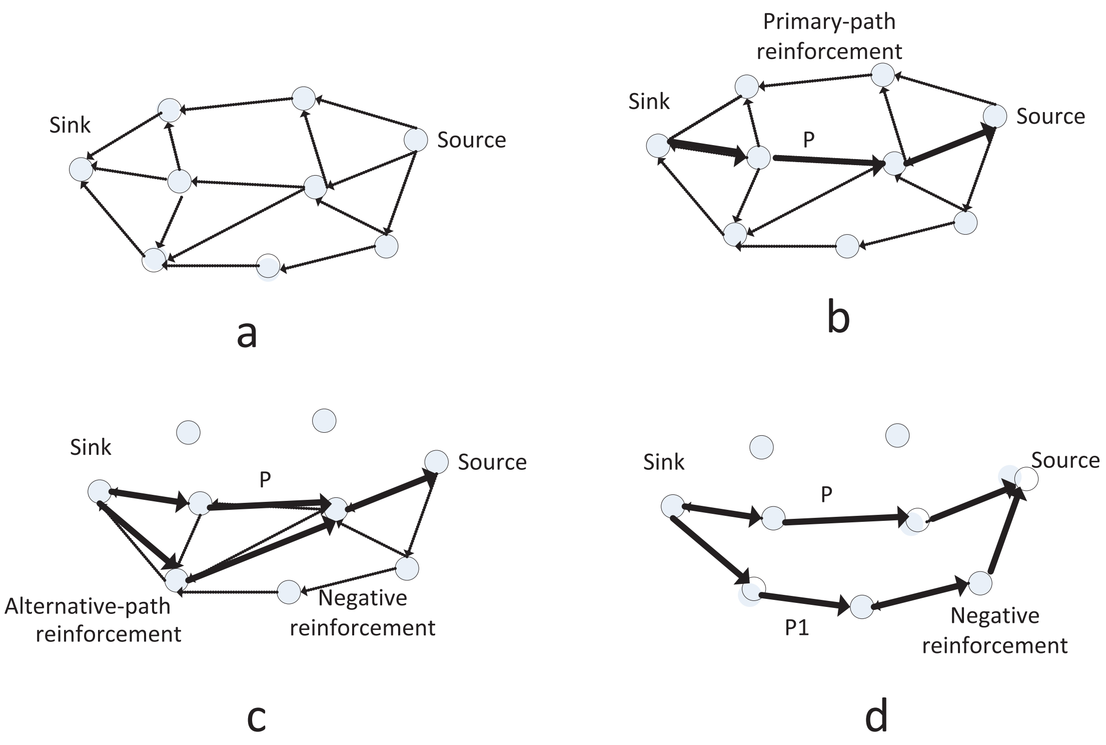


Figure 2.8: *Construct disjoint paths [49].*

neighbor chooses the second preferred neighbor that has not chosen before. If it has been chosen, it has to choose another one, as illustrated in Figure 2.8 (c). Finally, the $2-disjoint$ path is formed, as displayed in Figure 2.8 (d). Note that: (1) this method only uses the local information; (2) during the time setting up network to find multi-path, it might take a somewhat long time and consume too much energy.

The multi-path algorithms preferably to be distributed, e.g. [132] and [37]. [132] is a distributed algorithm to find out complete disjoined path. It is based on any other existing routing protocol to find the multi-path from the source to the destination, and then proposes a distributed algorithm to refine those paths in order to eliminate the paths which have mutual interference. [37] proposes a new routing protocol called GRAd, which is a routing protocol that makes no attempt to identify which of its neighbors is to relay a packet. It considers the mobility of sensor nodes. The nodes travel towards randomly chosen locations at random speeds, and pause for a fixed amount of time after reaching its destination. The simulation results show that GRAd is robust in the face of changing topology because it enlists multiple neighbor nodes to relay message from one place to another. Even if one node moves out of place, other nodes are often available to deliver the packet without resorting to rebuilding the route. However, there is a price needs to pay for this high robustness as [37] shows that the routing load is high. [135] proposes Resilient Routing Reconfiguration (R3), which consists of an off-line pre-computation phase and an online reconfiguration phase. By converting topology uncertainty into uncertainty in rerouting traffic, it applies linear programming duality to find the optimal route and protection route.

2.5 NP-completeness and NP-hard problems in resilient IoT

As mentioned in previous sections, many problems in IoT with regard to the resilient IoT turn out to be NP-completeness or NP-hard. A summary is made in this section. For NP-completeness and NP-hard problems [3], there is no polynomial-time algorithm to solve them. However, there are still some NP-completeness and NP-hard problems have provably bound on the solution, called $\alpha - ratio$ or $\alpha - approximation$ algorithm. $\alpha - approximation$ algorithm is a polynomial-time algorithm whose solution cost is at most α times the optimal solution cost. On the other hand, many related works have been focused on providing heuristics. However, heuristics could have arbitrarily poor performance, and do not guarantee provable running time like $\alpha - approximation$ algorithm.

If the problems are NP-completeness or NP-hard, usually it is proposed one heuristic algorithm, then its correctness is proven and $\alpha - approximation$ of its performance is presented if exists. If the algorithm needs to know the status of the entire network or have the physical (or logical) control center, it is called centralized algorithm, otherwise it is distributed algorithm. Table 2.1 lists many problems and algorithms regarding resilient IoT. Note that most algorithms proposed are centralized.

Table 2.1: *NP-completeness and NP-hard problems in resilient IoT.*

Ref.[1]	*Problems*	$\alpha - ratio$	*Complexity*	*Distr.*[2]
[56]	Minimize power while maintaining k-fault tolerance	$O(k)$ $O(k \log k)$	NA[3]	Yes
[103]	Find k-connectivity sub-graph with minimum cost	NA	NA	Yes
[86]	Find k pairwise vertex or edge disjoint paths in an acyclic directed network with length bounded by an integer. L is the sum of all edge	NA	$O(\|V\|^{k+1}L^{2k-2})$ $O(\|V\|^{k+1}L^{k})$	NO
[86]	Find two vertex or edge disjoint paths in a directed network such that maximum length of two path is minimized	at least 2	NA	NA

Continued on next page

Table 2.1 – continued from previous page

Ref.	*Problems*	$\alpha - ratio$	*Complexity*	*Distr.*
[120]	The reliability of the two disjoint-path connection equals to 1	NA	$O(\|E\| \log \|V\|)$	NO
[143]	Assign minimized transmission power to each sensor such that the induced topology containing only bidirectional links is strongly connected	$k^2 + O(k)$	NA	NO
[23]	Establish k-connectivity by placing additional sensors geographically between existing pairs of sensors with a minimum number of additional sensors	2	$O(\|V\|^2 \log \|V\|)$	NO
[31]	Place the minimum number of relay nodes for maintaining network connectivity such that the transmission range of each sensor does not exceed a constant	3, 2.5	$O(\|V\|^3)$	NO
[92]	Steiner tree problem with minimum number of Steiner points and bounded edge-length (STP-MSPBEL)	5	NA	NO
[72] [20]	Minimal number of relay nodes for 2-connected network	$O(D \log n)$ 10	$O((k\|V\|)^2)$[72]	NO
[78]	Find the smallest 2-connectivity weighted spanning sub-graph	$1 + \varepsilon$	NA	NO
[118] [84]	Find the minimum count and position of relay nodes	NA	$O(n^4)$[118]	NO
[2]	Minimum number of edges required to be added to obtain 2-vertex or 2-edge connected plane geometric graphs	NA	NA	NO

Continued on next page

Table 2.1 – continued from previous page

Ref.	*Problems*	$\alpha - ratio$	*Complexity*	*Distr.*
[140]	A minimum number of node placement for full k-connectivity and partial k-connectivity in the heterogeneous networks in terms of non-identical transmission range	$O(\sigma k^2)$ $O(\sigma k^3)$	NA	NO
[123]	directed k-strong connectivity with a minimal overall power assignment for uniformly spaced nodes on a line or planar space	$min(2, \frac{\Delta}{\delta})^{\alpha}$ $O(k^2)$	NA	NO

[1]Reference; [2]Distributed algorithm; [3]Not applicable.

2.6 Adopted models

A survey has been provided regarding various aspects of fault-tolerant network, including models, techniques and outcomes. This thesis has no intention to study every aspect of the network connectivity using all possible models, which would make the problem intractable. In Table 2.2, the models and methods that used in this thesis are outlined, in the same order that presented in this chapter.

Table 2.2: *The models adopted in the thesis.*

Models	**Models employed in the thesis**
Network model	Involve both heterogeneous and homogenous models
Transmission model	Involve both irregular and regular radio models
Asymmetric link	Only symmetric link
Failure model	Consider random and permanent node failure. Did not consider transient failure, link failure and faulty reading.
Deployment stages	Only apply to Post-deployment
Link quality	Not considered
Multi-path routing protocol	Not considered

2.7 Summary

Following the network design flow in Figure 2.1, the literature review is accomplished in this chapter (also see more details of our publication [156]). It provides a landscape, and assists the choice of appropriate models. It is evident that the notions, "Node degree" and "$k-connected$ network", are located at the core of the research of network connectivity. There are tremendous number of works have been done by various means: either from graph theory or networking point of view. The conclusions from the literatures help to better understand the network connectivity and fault-tolerant for large scale wireless networks.

What has missed in this chapter, but important one, is the self-adaptive software using software engineering approach. This becomes significant and needed when a real system is implemented. This topic will be covered in the implementation chapter, see Chapter 6.

Chapter 3

Connectivity Analysis Using a Probabilistic Model

As mentioned in previous chapters, whether a network is resilient to node failure, it can be evaluated from the network connectivity point of view. For example, if a network is $k - connected$, then it implies that the network can tolerate at most $k-1$ failure nodes, without losing the network connectivity. Certainly, it is implicitly assumed that all nodes in a network are required to be connected. Therefore, this chapter aims at analyzing the network connectivity over a very generic network, where nodes are randomly and uniformly deployed. This work provides insights into what is the probability that a network is $k - connected$, where $k = 1, 2, 3 \cdots$, and the number of nodes $n \to +\infty$, or loosely speaking n is big enough.

3.1 Network model

In this section, the following notations and definitions to facilitate the derivations are introduced:
n : total number of nodes deployed in target field, and $n \gg 1$.
A : area of node deployed.
ρ : node density, defined as $\frac{n}{A}$.
t : expected number of neighbors of node.
log : the logarithm to nature base e.

Next, the definitions that will state the starting point in the network model are introduced.

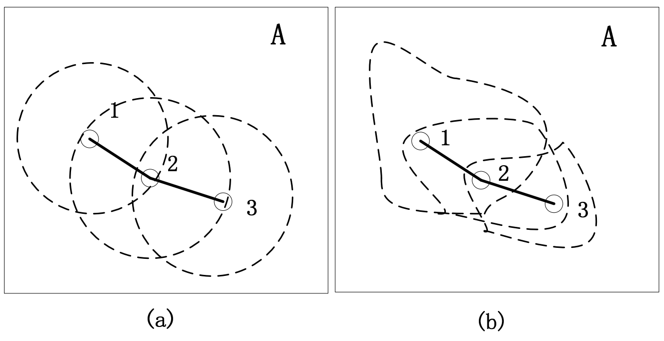


Figure 3.1: *(a) Disk communication model and (b) Irregular communication model.*

Definition 3.1.1. Node's *Communication range* is defined as the area where other nodes can receive its signal.

Definition 3.1.2. If the network is formed by n nodes randomly and uniformly deployed in area A, the network is called to follow *disc model*, and each node is modeled as a disk with radius r. The network is denoted by $S_{n,r}$.

This work adopts the disc model to describe the radio features of wireless communication, as shown in Figure 3.1(a). Specifically, the wireless communication range is drawn as a perfect circle with radius r. However, in practice the radios can hardly be an ideal circle (e.g. Figure 3.1(b)), but it will be shown in the analysis that the shape of the communication range does not matter, as long as the area of the communication range covered is the same, then the effective communication range is the same as it is a circle. On the other hand, what really matter is whether the communication range of each node is the same. So, this thesis has two sections, Section 3.2 and Section 3.3, to show different results.

In general, there are analytic results if all nodes have the same communication range, which is named homogenous deployment; while the analytic results do not exist if all nodes do not have the same communication range, which is called heterogeneous deployment, this work performs numerical simulation.

3.2 Homogenous node deployment

The probability model when each node has the same communication range in network $S_{n,r}$ is presented in this section; besides, the details of derivations are given, and the results are validated by computer simulation.

3.2.1 Network connection probability analysis

For randomly and uniformly distributed nodes with density ρ, the number of nodes in the area πr^2 has a *Poisson* distribution with mean $\rho\pi r^2$ [19]. Furthermore, the probability of a node having N neighbor nodes in relation to the communication range r is given by:

$$P(N) = \frac{(\rho\pi r^2)^N}{N!} e^{-\rho\pi r^2}. \tag{3.1}$$

Therefore, the probability of $S_{n,r}$ being *k-connected* can be computed through:

$$P(k) = (1 - \sum_{N=0}^{k-1} \frac{(\rho\pi r^2)^N}{N!} e^{-\rho\pi r^2})^n. \tag{3.2}$$

Further, let:

$$t = \pi r^2, \tag{3.3}$$

t can be interpreted as the area that the communication range covered. The equation (2.2) is only a special case, in which the area is a disc. It is not necessary a disc, though. If $\rho = 1$, t actually is the expected number of neighbors a node has. Whether ρ is equal to 1 is irrelevant in this model, because if ρ is not 1, say ρ', then let $r' = \frac{r}{\sqrt{\rho'}}$. Another interpretation is that keeps the average number of nodes on unit area being 1, but what really matter is the communication range r impacts on the network connection probability. Therefore, without loss of generality, it is assumed that $\rho = 1$. In order to simplify denotation, define:

$$W(t,k) = \sum_{N=0}^{k-1} \frac{(t)^N}{N!} e^{-t}. \tag{3.4}$$

So, (3.2) can be re-written as:

$$P(t,k) = (1 - W(t,k))^n, \tag{3.5}$$

and the corresponding first derivative w.r.t. t is:

$$\begin{aligned} \frac{dP(t,k)}{dt} &= -n(1 - W(t,k))^{n-1} \frac{dW(t,k)}{dt} \\ &= n(\frac{e^t - \sum_{N=0}^{k-1} \frac{t^N}{N!}}{e^t})^{n-1} (e^{-t} \frac{t^{k-1}}{(k-1)!}). \end{aligned} \tag{3.6}$$

The function $P(t,k)$ can be written as $P(t) = (1 - e^{-t})^n$ when $k = 1$, namely $1 - $ *connected* network.

Before proceeding to the formal analysis, Figure 3.2 draws the equation 3.5 when $k = 1$, which is the probability that the network connectivity being $1 - $ *connected* with various number of nodes. It can be seen that the functions have very similar shape, and take the same amount of t to grow from near 0 to reach 1. In what follows, this thesis will prove those intuitive conclusions hold.

In order to prove the hypothesis, it is needed to prove the following proposition:

Proposition 3.2.1. *Let* $P(t) = (1 - e^{-t})^n$ *and* $n \gg 1$*, then following statements hold:*

1. *For every* $0 < \varepsilon < (1 - \frac{1}{n})^{(n-1)}$*, there exists* $0 < t_1 < t_2$ *such that* $P'(t_1) = P'(t_2) = \varepsilon$*.*

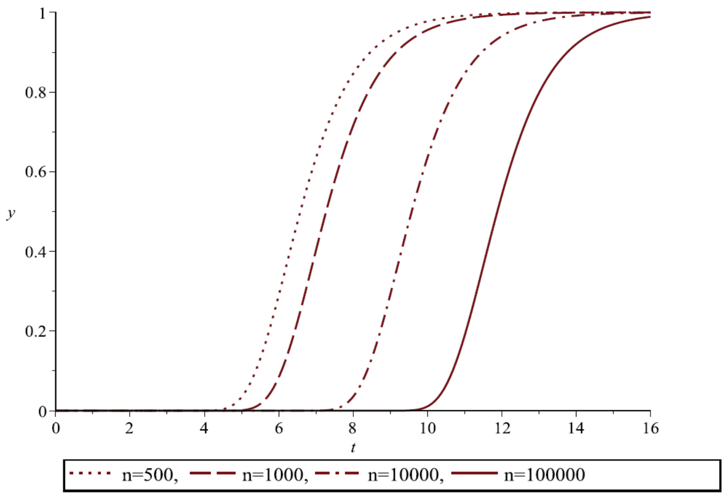


Figure 3.2: *Network connection probability $P(t)$ when $n = 500, 1000, 10000, 100000$.*

2. *$P(t)$ has a flex point at $(\log(n), (1-\frac{1}{n})^n)$.*

Proof.

1. First, Figure 3.3 (left) shows (3.5) is a monotonically increasing function. Strictly, $e^t = \sum_{N=0}^{+\infty} \frac{t^N}{N!}$ because it is the Taylor expansion of e^t, and $t \geq 0$. So, (3.6) is greater than 0, which leads to the function (2.1) a monotonically increasing function for any $k \geq 1$. Consequently, $P(t)$ is monotonically increasing function w.r.t. t.

   Then, consider the first and second derivative functions of $P(t)$ w.r.t. t:

$$P'(t) = n(1-e^{-t})^{n-1}e^{-t}, \tag{3.7}$$

$$P''(t) = ne^{-t}(1-e^{-t})^{n-2}(ne^{-t}-1). \tag{3.8}$$

   It is evident that $P''(t)$ vanishes at $t = 0$ and $t = \log(n)$. Furthermore, $P''(t) > 0$ for $t \in (0, \log(n))$ and $P''(t) < 0$ for $t \in (\log(n), \infty)$. Therefore, $P'(t)$ is an increasing function in the interval $(0, \log(n))$ and a decreasing function in $(\log(n), \infty)$. Hence, $P'(t)$ reaches the maximum value at $t = \log(n)$ (see Figure 3.3 (up)). On the other hand, since $\lim_{t\to\infty} P'(t) = 0$, by applying *Bolzano* Theorem, for every $0 < \varepsilon < P'(\log(n)) = (1-\frac{1}{n})^{(n-1)}$ there exists $0 < t_1 < t_2$, such that $P'(t_1) = P'(t_2) = \varepsilon$. For example, as shown in Figure 3.3 (down), $\varepsilon = (1-\frac{1}{500})^{(500-1)} \approx 0.368$.

2. It is derived from the proof of statement above. $\square$

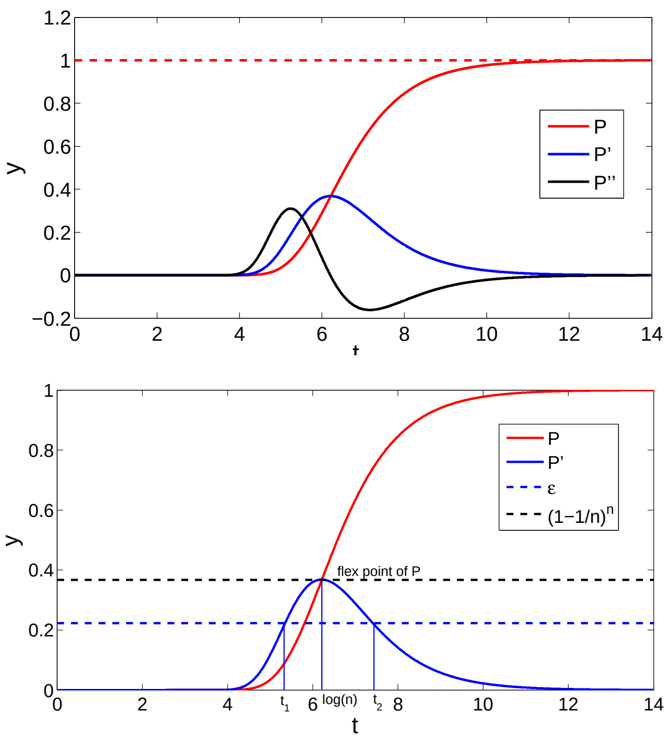


Figure 3.3: Network connection probability function $P(t)$, and corresponding $P'(t)$ and $P''(t)$ (up); values t_1 and t_2 for a given ε and $n = 500$ (down).

Remark 3.2.2. Note that $0 < \varepsilon < (1 - \frac{1}{n})^{(n-1)} < 1$ since $n \gg 1$.

To further analyze $P(t)$, the previous Proposition can be applied to obtain the following theorems.

Theorem 3.2.3. *Let $P(t) = (1 - e^{-t})^n$, s.t. $n \gg 1$, $b > 1$ and $\varepsilon = e^{-b}$, then there exists a constant number of neighbors, $C(\varepsilon) = log(\frac{1-log\varepsilon}{\varepsilon})$, for which the network becomes connected with probability increasing from $P(t_1 = \log(\frac{n}{b+1}))$ to $P(t_2 = \log(n) + b)$.*

Proof. First, it can be observed that $\varepsilon = e^{-b}$ satisfies the hypothesis in Proposition since $\varepsilon = e^{-b} < e^{-1}$, therefore $0 < \varepsilon < (1 - \frac{1}{n})^{(n-1)}$. Let $x = e^{-t}$ and define $f(x)$ as:

$$f(x) = n(1 - x)^{n-1}x - \varepsilon. \tag{3.9}$$

Then, the goal is to find the roots of (3.9). For this purpose, considering the corresponding derivative function:

$$\begin{aligned} f'(x) &= n(1 - x)^{n-1} - n(n - 1)(1 - x)^{n-2}x \\ &= n(1 - x)^{n-2}(1 - nx). \end{aligned} \tag{3.10}$$

Note that $f(x)$ is the function $P'(t) - \varepsilon$ under the change of variable $x = e^{-t}$, and by applying the Proposition there exists only two roots, say x_1 and x_2 s.t. $0 < x_2 <$

$\frac{1}{n} < x_1 < 1$. *Newton* method can be used to find out the approximation of roots x_1 and x_2. Let x_0 be the initial value, according to (3.9) and (3.10), yield:

$$\begin{aligned} x &\approx x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= x_0 - \frac{n(1-x_0)^{n-1}x_0 - \varepsilon}{n(1-x_0)^{n-2}(1-nx_0)} \\ &= x_0 - \frac{n(1-x_0)x_0 - \frac{\varepsilon}{(1-x_0)^{n-2}}}{n(1-nx_0)} \\ &= x_0 - \frac{(1-x_0)x_0}{1-nx_0} + \frac{\frac{\varepsilon}{(1-x_0)^{n-2}}}{n(1-nx_0)}. \end{aligned} \tag{3.11}$$

Additionally, let $x_0 = 0$ as the initial value to approximate x_2 and $x_0 = \frac{b}{n}$ (where $b > 1$) as the initial value to solve x_1.

$$x_2 = \frac{\varepsilon}{n} = \frac{e^{-b}}{n}, \tag{3.12}$$

$$\begin{aligned} x_1 &= \frac{b}{n} - \frac{(1-\frac{b}{n})\frac{b}{n}}{1-b} + \frac{\frac{\varepsilon}{(1-\frac{b}{n})^{n-2}}}{n(1-b)} \\ &= \frac{1}{n}\left(\frac{\frac{b^2}{n} - b^2 + \frac{\varepsilon}{(1-\frac{b}{n})^{n-2}}}{1-b}\right). \end{aligned} \tag{3.13}$$

Taking into consideration the fact that $(1-\frac{b}{n})^{n-2} \to e^{-b}$ when $n \to +\infty$ and $\varepsilon = e^{-b}$, therefore:

$$\frac{\frac{b^2}{n} - b^2 + \frac{\varepsilon}{(1-\frac{b}{n})^{n-2}}}{1-b} \to 1+b, \tag{3.14}$$

written $b = -\log(\varepsilon)$, and let $t_2 = -\log x_2$, $t_1 = -\log x_1$, and define $C(\varepsilon) = t_2 - t_1$. It can be concluded that (as $n \to +\infty$):

$$C(\varepsilon) = \log(\frac{x_1}{x_2}) = \log(\frac{\frac{1}{n}(\frac{\frac{b^2}{n}-b^2+\frac{\varepsilon}{(1-\frac{b}{n})^{n-2}}}{1-b})}{\frac{\varepsilon}{n}}) \to log(\frac{1-log(\varepsilon)}{\varepsilon}). \tag{3.15}$$

Finally, because (3.12) has $t_2 = \log(n) + b$, so $t_1 = t_2 - C(\varepsilon) = \log(\frac{n}{b+1})$. Since $P(t)$ is an increasing function, it concludes that the network becomes connected with probability increasing from $P(t_1 = \log(\frac{n}{b+1}))$ to $P(t_2 = \log(n) + b)$. □

Moreover, more results can be obtained when $t = t_1$ and $t = t_2$, by applying some basic derivations from the model.

Theorem 3.2.4. *Let $\varepsilon = e^{-b}$ and $b > 1$, if $\pi r^2 = logn + b$ then network connection probability $P(S_{n,r}$ connected$)$ is at least $1 - \varepsilon$ when $n \to +\infty$.*

Proof.

$$P(S_{n,r} \textit{ connected}) = (1 - \frac{e^{-b}}{n})^n \geq 1 - n\frac{e^{-b}}{n} = 1 - \varepsilon \tag{3.16}$$

□

Remark 3.2.5. This theorem shows that as $b \to +\infty$ (equivalently, $\varepsilon \to 0$), the network connection probability tends to 1, meanwhile the network has degree $\log n + b$. The literature [110] proves that if a network does not have any links at the beginning, and later links are added to connect nodes, the resulting network becomes *k-connected* as soon as network degree is k. Therefore, this theorem shows that once network becomes connected, it turns out to be (log n + b)-connected with high probability. This conclusion is consistent with the result in [16]: by increasing k network becomes *s-connected* very shortly after it becomes connected, for $s = O(\log n)$. (log n + b)-connected network makes the IoT more resilient against node failure, because there are *log n + b* distinct paths from one node to any other nodes.

Theorem 3.2.6. *Let $\varepsilon = e^{-b}$ and $b > 1$, if $\pi r^2 = \log \frac{n}{b+1}$ then the network connection probability $P(S_{n,r}$ connected$)$ is about 0.36ε when $n \to +\infty$.*

Proof.

$$P(S_{n,r} \textit{ connected}) = (1 - \frac{b+1}{n})^n \to e^{-b-1} = 0.36e^{-b} = 0.36\varepsilon. \tag{3.17}$$

□

Theorem 3.2.7. *Let $\varepsilon = e^{-b}$ and $b > 1$, if $\pi r^2 = \log n + b$ then the network connection probability is $e^{-e^{-b}}$, when $n \to +\infty$.*

Proof.

$$P(S_{n,r} \textit{ connected}) = (1 - \frac{e^{-b}}{n})^n \to e^{-e^{-b}}. \tag{3.18}$$

□

Remark 3.2.8. This conclusion is the same as [17] and has similar form in the Erdös-Rényi random graph [60].

Corollary 3.2.9. *Let $0 < \varepsilon < e^{-1}$, if the probability of network being connected is 0.36ε when communication range is πr^2, then by increasing node communication range by constant $C(\varepsilon) = \log(\frac{1-\log\varepsilon}{\varepsilon})$, namely $\pi r^2 + C(\varepsilon)$, the probability of network being connected is at least $1 - \varepsilon$.*

Proof. The conclusion can be drawn from the previous theorems above. □

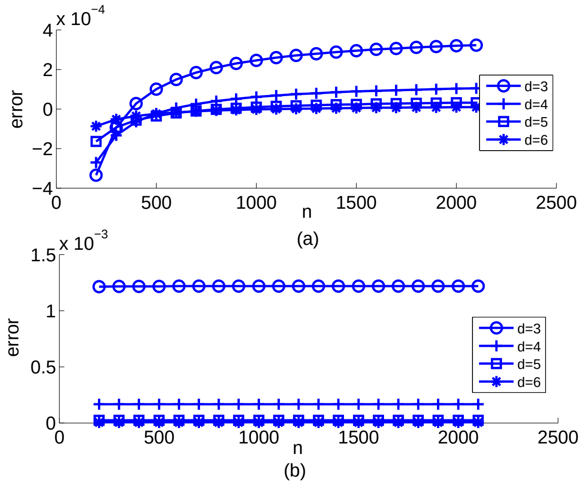


Figure 3.4: *(a) Errors of Theorem 3.2.4; (b) Errors of Theorem 3.2.6.*

Remark 3.2.10. The corollary provides a guide on control the communication range so that the network is connected with high probability. For example, in the case that the probability is very low due to the communication range is too short, so one further wants to know how large should increase communication range to get a much high probability. Surprisingly, the corollary shows that the incremental is a constant for any size of network, in the condition that the size of network is large enough, e.g. 500 nodes. It implies that, at certain point, further increasing the communication range would be pointless and waste energy, as it is no longer to help too much to increase the probability that the network being connected.

3.2.2 Validation

The analysis in the previous Section 3.2.1 is asymptotical, because it theoretically requires $n \to +\infty$. This section validates the results by simulation, which shows that the theories still hold when n is big enough, e.g. 500, with acceptable errors. The error of Theorem 3.2.4 is defined as $(1 - \frac{e^{-b}}{n})^n - (1 - e^{-b})$, and the error of Theorem 3.2.6 is defined as $(1 - \frac{b+1}{n})^n - 0.36e^{-b}$. To validate Theorem 3.2.4 and Theorem 3.2.6, Figure 3.4 calculates the error between theoretical results and approximation values with different n and b. The errors for both theorems are very small, which indicate that both have a good approximation.

Table 3.1 illustrates that the values of x_1, x_2, t_1, t_2, $C(\varepsilon)$ and corresponding values of $P(t_1)$ and $P(t_2)$, for $n = 500, 1000, 10000, 100000$. For any n in the table, the obtained value $t_2 - t_1 \approx C(\varepsilon) = 5.60944$. Of course, in a real network the number of neighbors has to be an integer, so 6 more neighbors can make sure that the network is connected with very high probability. In other words, if the area covered is wider, certainly the communication range of node required to make sure the network being connected with

Table 3.1: *Connection probability with different size of network.*

n	t_1	t_2	$t_2 - t_1$	$P(t_1)$	$P(t_2)$
500	4.60517	10.21461	5.60944	0.65705%	98.18507%
1000	5.29832	10.90776	5.60944	0.66540%	98.18509%
10000	7.60090	13.21034	5.60944	0.67295%	98.18511%
100000	9.90349	15.51293	5.60944	0.67371%	98.18511%

high probability is higher. But the most important conclusion can be drawn from the theory is: starting from the network being connected with very low probability(at around 0.67%), by adding 6 more nodes in the neighbor list, the whole network becomes connected with very high probability (at around 98%), and 6 is a "magic number" for any size of the network. Further, once the network is connected, it soon becomes (log n + 6)-connected network, which can tolerate at most (log n + 5) nodes failures at the same time.

3.3 Heterogeneous node deployment

This section evaluates the same problem but with a more practical model, which the communication ranges of each node are not identical. Rather, they have different communication range. In order to model heterogeneous communication range, this study applies the stochastic process called Cox process. Unlike the Section 3.2, there is no analytical results, hence the computer simulation is leveraged instead.

3.3.1 Cox process model

Cox process is a doubly stochastic Poisson process, where the parameter of Poisson process itself is another stochastic process.

Formally, given a random variable X follows Poisson distribution with parameter λ, written as $X \sim Poisson(\lambda)$ or $Poisson(X = k; \lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$, where the parameter λ is defined by another distribution N, namely $\lambda \sim N$.

Recall that in the equation (3.1), the number of neighbors is defined as random variable N, and $N \sim Poisson(\rho\pi r^2)$. Now, the problem of this thesis is formalized into a more general case where the communication ranges of each node are not necessary identical, instead it is i.i.d. normal distribution. To do this, it is assumed that the communication range of each node is i.i.d. random variable, and $\rho\pi r^2 \sim Norm(\mu, \sigma^2)$. Again, let the node density $\rho = 1$, and the number of neighbors of a node, denoted as k_t, so $k_t \sim Norm(\mu, \sigma^2)$.

3.3.2 Network connection probability analysis

By convention, $E[V]$ denotes the expected value of a random variable V. The expected neighbors of node can be computed by using the "Law of total expectation":

$$E[k_t] = E[E[k_t|t]] = E[t] = \mu. \tag{3.19}$$

For $k \geq 1$, the probability of node i having at least k neighbors is given as:

$$P_{i,k} = g_i(t_i) = 1 - \sum_{N=0}^{k-1} \frac{t_i^N}{N!} e^{-t_i}. \tag{3.20}$$

For a special case, when $k = 1$, $P_{i,1} = 1 - \frac{1}{e^{t_i}}$ is the probability that node i is not isolated. The expectation of $E[P_{i,1}]$ and variance $Var[P_{i,1}]$ are not difficult to calculate. They are:

$$E[P_{i,1}] = 1 - e^{-\mu + \frac{1}{2}\sigma^2}. \tag{3.21}$$

$$Var[P_{i,1}] = (e^{\sigma^2} - 1)(E[P_{i,1}])^2. \tag{3.22}$$

For $k > 1$, the distribution of $P_{i,k}$ does not have a closed-form expression. If n is big enough, the probability of network being $k - connected$ is approximated as follows:

$$P_k = \prod_{i=1}^{n} P_{i,k}. \tag{3.23}$$

Since parameter t is a random variable, P_k is a random variable as well. Let $P_{min} = \min\{P_{1,k}, P_{2,k}, \cdots, P_{n,k}\}$, because $0 \leq P_{i,k} \leq 1$ so

$$P_{min}^n \leq P_k \leq P_{min}. \tag{3.24}$$

Therefore the obstruction of connection probability of entire network is the node which has the minimal communication range.

3.3.3 Numerical simulations

Due to the complexity of the model, there are no analytical results, so this thesis conducts the computer simulation to evaluate the network when the node communication range is heterogeneous.

The theorem 3.2.4 from Section 3.2 is used to guide the choice of the mean for the normal distribution to generate random communication ranges. When $b = 5$, the probability of the network being connected is at least 99.33%, which is high. So, $\log n + b = \log n + 5$ as average μ of communication range; for instance, if $n = 500$, then $\mu = 11.2$. Of course, it can choose other μ, either higher or lower, but the connection probability of the network would be extremely high or low, which would not provide us much valuable information. Besides, in the following simulation, various variances

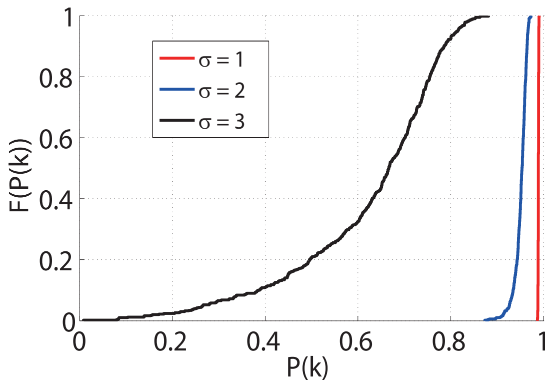


Figure 3.5: *Cumulative distribution function (CDF) of P_k when $\sigma = 1, 2, 3$, $n = 500$, and $k = 1$.*

will be configured to add the diversity of the communication ranges. The following probabilistic problems are investigated:

(1) cumulative distribution function (CDF) of P_k with various σ and k. The CDF of P_k is calculated after 500 runs, as shown in Figure 3.5 and Figure 3.6;

(2) given μ and σ, what is the probability of network being $k - connected$ as the number of nodes deployed grows. This is done by computing average of $P_{n,k}$ after 500 runs for a given number of nodes, as illustrated in Figure 3.7 and Figure 3.8;

(3) How to choose the parameters in order to get the required connection probability. This will be discussed in Section 3.3.4.

In Figure 3.6 $\sigma = 3$ and $k = 2$ (black line), the probability of network being connected is almost sure less than 40%; whereas, the probability of network being connected is almost sure more than 40% when $\sigma = 2$ and $k = 2$ (blue line). This implies that higher variance in the communication range, which naturally leads to the lowest communication range is more likely, results in lower possibility that the network being connected.

Moreover, Figure 3.7 illustrates the connection probability as N nodes deployed in network when $\sigma = 2$ and $k = 1, 2, 3, 4$. Figure 3.8 shows the changes when $\sigma = 1$, 2, 3 and $k = 1$. Both figures show that the average of P_k is the decreasing function of k, N and σ. The network connection probability as network size growing becomes predictable. For instance, Figure 3.8 shows the network average connection probability for $\sigma = 3$ is about 73% when the network has 250 nodes, but the probability falls to 55% if the network size is doubled. Figure 3.7 shows how resilience performance decreases when network size grows or the probability decreases if higher resilience performance is required. For example, for networks have 200 nodes, the probability that this network can tolerate 1, 2, 3 (because $k = 2, 3, 4$) nodes failure are about 83%, 50%, 10%, respectively.

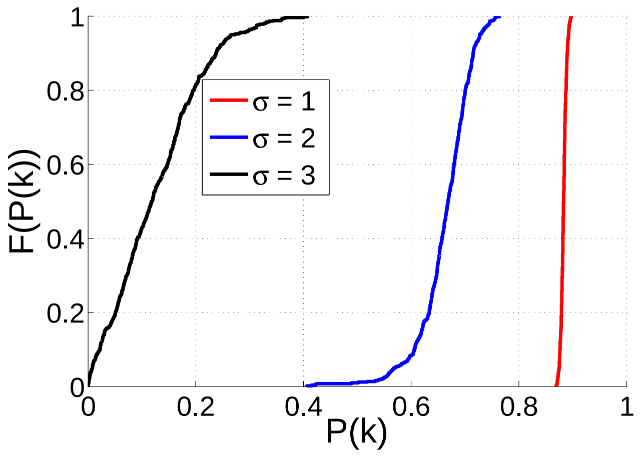


Figure 3.6: *Cumulative distribution function (CDF) of P_k when σ = 1, 2, 3 and n = 500, k = 2.*

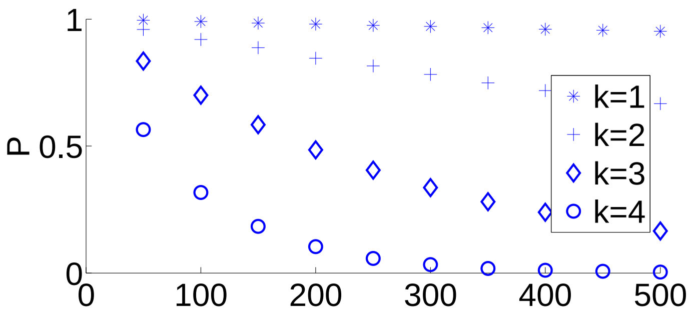


Figure 3.7: *k − connected probability P when σ = 2, and k = 1, 2, 3, 4.*

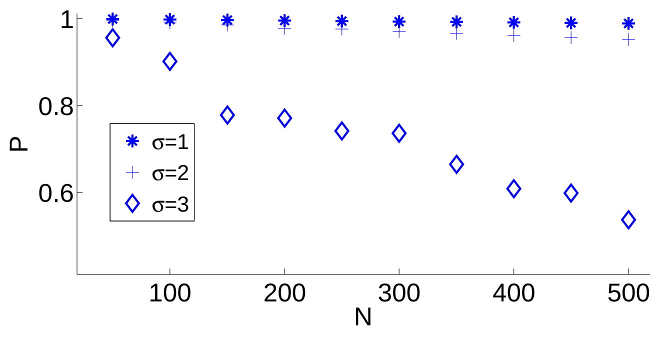


Figure 3.8: *Connection probability P when σ = 1, 2, 3 and k = 1.*

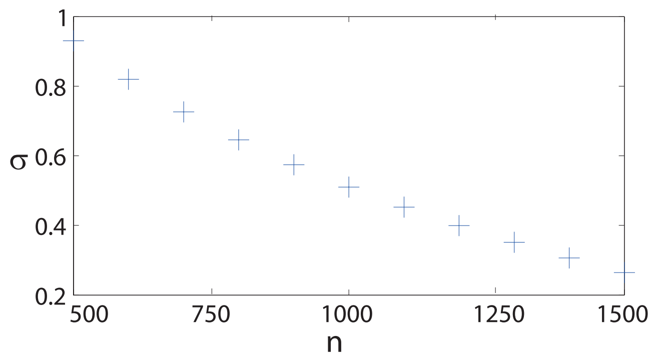


Figure 3.9: *Required σ for 90% connection probability when $\mu = 10$.*

3.3.4 Choose distribution parameter

Sometimes, it is more interesting to study what are the distribution parameters can maintain the given P_k. For instance, the network simulator (e.g. NS2) wants to choose appropriate parameters to simulate real networks. It is desirable to keep most of nodes in a network being connected, meanwhile remains the nodes reasonably heterogeneous.

Based on the equation (3.6), the function (3.20) is a monotonically increasing function w.r.t. $t_i \in [0, +\infty)$, and its inverse function is written as:

$$t_{i,k} = g_i^{-1}(p_{i,k}). \tag{3.25}$$

Let $p_{i,k}^{(0)}$ be a instance of $p_{i,k}$, thus $t_{i,k}^{(0)} = g_i^{-1}(p_{i,k}^{(0)})$. The probability $p_{i,k}$ being greater than $p_{i,k}^{(0)}$ is given by:

$$\int_{t_{i,k}^{(0)}}^{+\infty} f_T(t)dt, \tag{3.26}$$

where $f_T(t)$ is the probability density function of t. If the probability of a network required to keep network $k-connected$ is at least P_0, the corresponding probability for each node is at least:

$$p_{min} \geq P_0^{\frac{1}{n}}. \tag{3.27}$$

With formula (3.25)-(3.27), the required density function parameter of communication range for given P_0 can be calculated.

In the following, the usage of the theory with an example is explained. Assume that one wants to design a network to be well connected. For instance, there are 500 nodes are deployed in $\sqrt{500} \cdot \sqrt{500}\ m^2$, and the communication area t follows the distribution $t \sim N(\mu, \sigma^2)$ with mean $\mu = 10m$, but the standard deviation σ is unknown. Besides, it is expected that the desired probability of the whole network being $k - connected$ is at least 90%. Now, one wants to know the standard deviation of this distribution. For $k = 1$, according to (3.25) and (3.27), corresponding minimal range is $t_{i,1}^{(0)} = 8.46$. In order to make the probability of $t_{i,1}$ greater than $t_{i,1}^{(0)}$ is high, e.g. at least 95%, according to (3.26) $t_{i,1} = \mu - 1.65\sigma$. Therefore corresponding standard deviation σ should be no more than 0.93. This is useful in the case that one uses network simulator to choose appropriate parameters to design high probability connected networks. Figure 3.9 shows the required σ in order to make the network connection probability at least 90% when the number of nodes are different.

3.4 Summary

This chapter investigates the probability of all nodes being $k-$*connected* in a randomly and uniformly deployed network, and all results have been published in [66]. The thesis considers two distinct circumstances: the communication range of each node is the same, or not. In the former case, the asymptotical results are obtained; in the latter case, the process is modeled as a Cox process, and evaluated by computer simulation.

The results in this chapter can be applied to the situation where network designers want to know how to design a resilient network that can tolerate $k-1$ nodes failure; or, given a network, what is the probability that a network can be $k-$ *connected*. However, the results in this chapter rely on the assumptions that the nodes are uniformly deployed, and the number of nodes has to be big enough (theoretically, goes to infinity). Nevertheless, this chapter gives insights into the network fault-tolerant capability in a broad sense.

This chapter lays the theoretical foundations for the rest of the thesis. The remaining works are to develop some techniques that can control the ND, in turn control network connectivity and improve fault-tolerance capability of a network.

Chapter 4

Maintain the IoT Resilience Using Fuzzy-logic Control

The previous chapter analyzed the network connection probability in the condition that all nodes are randomly and uniformly deployed. The results reveal the relevance between the network connectivity probability and the communication range of nodes. This chapter leverages the fuzzy-logic approach to control the communication range (or, transmission power), in turn control the node degree, which leads to the network connected with high probability. On the other hand, the connected network is able to tolerate certain amount of node failures, according to the theory in the previous chapter.

There are mainly two approaches to design the fuzzy-logic controller, learning-based or heuristics. Each has its benefits and drawbacks. This thesis shows both approaches separately in Section 4.2 and Section 4.3. The simulation results and analysis will be presented in details in Section 4.5 and Section 4.6. An approach that can optimize the control system parameters will be presented in next chapter, in Section 5.2.

4.1 Fuzzy-logic control theory

Fuzzy control, or fuzzy-logic control, was first introduced by [159], and has inspired many applications, in particular when the control system are very complex and dynamic. In the past decades, fuzzy control is one of the most active areas of research. Fuzzy logic is much closer inspirit to human thinking than other traditional logic control system [82]. The inputs and outputs of traditional control system are either crisp (e.g. numerical value 3, 45, 23.5, etc.), or binary (e.g. 0 or 1, true or false). In contrast, the inputs

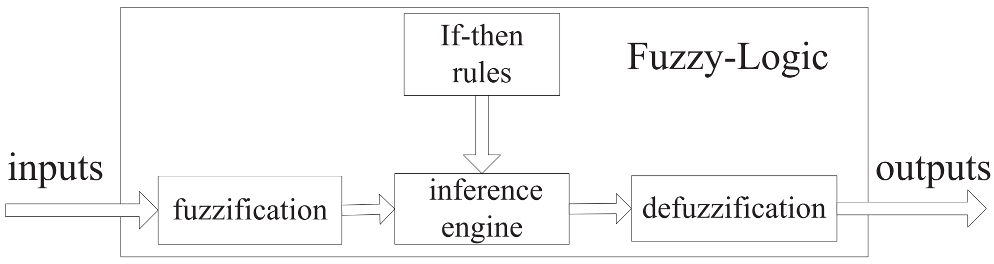


Figure 4.1: *Fuzzy-logic control system.*

and outputs of fuzzy logic, basically, are more like human language to describe the uncertainty of the real world. For instance, human being tends to use the words "high", "low" or "a little high" to describe the water temperature, rather than exact value, like 90 degree or 40 degree. So, the fuzzy logic can be viewed as the bridge between the precise mathematical control and human-like decision making [82].

The fuzzy control brings many benefits that traditional control does not have. First, it is easier to model the uncertainty of control system when it is very complex, and dynamic. It is very likely that there are not any precise mathematical models to describe the system, where traditional control approach may fail. Second, it is feasible to convert the expertise knowledge of domain to a control system quickly. The domain expertise may be not able to design a control system, but they are able to describe the behaviors of the system by using their own language. Hence, their knowledge can be translated to a control system by using the fuzzy control. In a large distributed IoT, the network is extremely complex and time-varying. Therefore, the fuzzy-logic control is a very good technique to solve some problems in IoT.

Figure 4.1 shows a typical fuzzy-logic control system. "fuzzification" transforms a crisp input variable (e.g. node degree in this case) into a linguistic variable, e.g. High, Medium, Low; "inference engine" maps the linguistic inputs onto linguistic outputs based on "if-then" rules; "defuzzification" converts the linguistic outputs of inference engine to crisp variables. Both linguistic input and output are represented by the membership functions. For complete fuzzy logic theory, one can refer to [76].

The key components of the fuzzy-logic are the inference engine (if-then rules) and the membership function. Fortunately, there are some techniques can learn those if-then rules and membership functions automatically. One of the most efficient ways is the Neural Network (NN), which also is well developed in machine learning field. The back prorogation (BP) algorithm is used to estimate the membership function. However, like any other machine learning algorithms, the training data must be provided, which is not always an easy task. On the other hand, the inference engine and membership function can be constructed by domain experts. However, people usually end up with spending a lot of time on tuning the parameters of the membership functions, and if-then rules. This chapter considers both ways to design fuzzy-logic controller. In addition, another

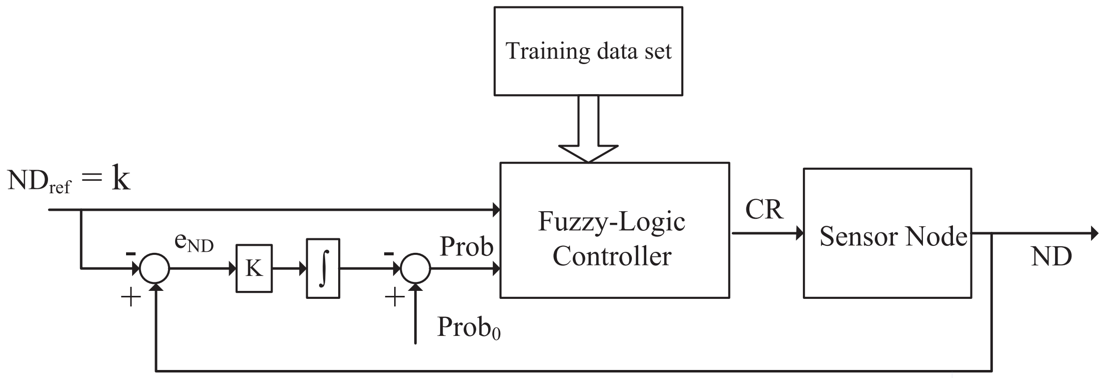


Figure 4.2: *Learning-based Fuzzy-logic Topology Control system (LFTC).*

way to ease the tedious works of tuning parameters is to combine other techniques with fuzzy-logic, e.g. PID control. This chapter also involves this approach, and will further discuss the parameters optimization in the control system.

4.2 Fuzzy-logic control based on learning algorithm

First of all, the notions that will be use in this chapter are introduced.

- ND: node degree or number of neighbors a node has.
- ND_{ref} or k: reference ND or desired ND. Sometimes, also use k to represent ND_{ref}.
- ND_{lost}: number of lost neighbor(s).
- e_{ND}: node degree error $e_{ND} = ND - ND_{ref}$.
- CR: communication range.
- E: node residual energy.
- BS: base station.

Figure 4.2 shows the first system designed based on the learning algorithm, called Learning-based Fuzzy-logic Control (LFTC).

Adjusting the communication range is a very common feature in many sensor nodes, e.g., Sun SPOT sensors. It is worth noting that in many sensor platforms, the only parameter allowed to change is the transmission power, rather than directly change the communication range. In practice, the range in radio communication is generally described by equation 4.1, called Friis equation:

$$P_R = P_T \frac{G_T G_R \lambda^2}{(4\pi)^2 d^n} \tag{4.1}$$

where,

- P_R: Power available from receiving antenna
- P_T: Power supplied to the transmitting antenna
- G_R: Gain in receiving antenna
- G_T: Gain in transmitting antenna
- λ: wavelength, where $\lambda = \frac{c}{f}$, f=frequency, e.g. the standard IEEE 802.15.4 wireless communication operating on 2.4GHz, 915MHz,868MHz
- d: Distance
- c: Speed of light in vacuum $299.972458 \cdot 10^6 m/s$
- n: Wireless environment, e.g. in free-space, n = 2; in building line of sight, $n \in [1.6, 1.8]$; in obstructed building $n \in [4, 6]$, etc.

The Friis equation can predict the signal degradation for a given transmission distance and environment. More practical, the power transmission is discrete, rather than continuous. For instance, the widely used wireless standard IEEE 802.15.4, defined the transmission power can be changed from -31dBm to 31dBm, with total 63 levels. Besides, the communication range is irregular, rather than a circle. It is already discussed in more details in Section 2.2.2. However, there are various simple models for the purpose of simulations, e.g. the one used in the Section 4.5 to describe the transmission range and power consumption.

This chapter mainly targets on the computer simulation to evaluate the proposals for a large and generic network, while Chapter 6 will study a more complex but a real world scenario that will be applying a self-adaptive control system in Sun SPOT sensor nodes. Due to the limited number of sensor nodes in the lab, the control system has to be modified and not all proposals in this chapter will be evaluated, only the one which requires the lowest computational resources will.

The output of LFTC is CR. Since the target is to reach a specific ND for LFTC, one of the inputs is ND_{ref} or k. Again, note that ND_{ref} and k are use interchangeably, because ND_{ref} is used in control systems and k often represents the node degree in the graph theory. Chapter 3 concluded that k turns out to be a very important metric that tells how well the network being connected, and how many nodes failure can be tolerated before a network disconnected.

For a random and uniform deployment, the probability that the node degree is at least bigger than k is given by equation (1.2) (also, see [19]). For the reason of completeness of this section, it is re-wrote here as equation (4.2), where r is the node

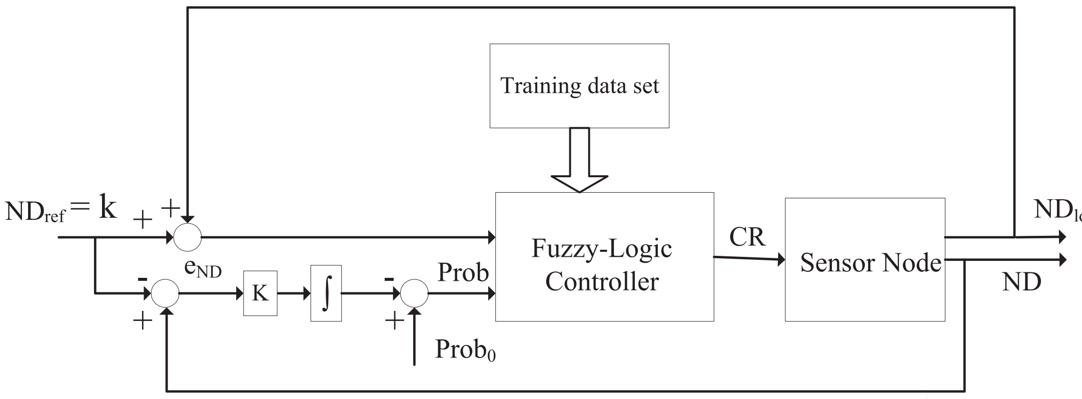


Figure 4.3: *Alternative: Learning-based Fuzzy-logic Topology Control system (LFTC).*

communication range, n is the total number of nodes in the field, node density $\rho = \frac{n}{A}$, and A is the area of a deployment field.

$$P(ND \geq k) = f(k, r) = 1 - \sum_{N=0}^{k-1} \frac{(\rho\pi r^2)^N}{N!} e^{-\rho\pi r^2} \tag{4.2}$$

Therefore, the probability that a node has degree k is another fuzzy-logic controller input, denoted by $Prob$. In practice, k is an integer and communication range has an upper bound r_{max}, so integer $k > 0$ and $0 \leq r \leq r_{max}$. The probabilistic results serve as the prior knowledge to generate training data set, which will be discussed soon at Section 4.2.1.

Furthermore, taking into consideration that the node failures at runtime, the fuzzy inputs are $ND_{ref} + ND_{lost}$ and the probability $Prob$ that a node has $ND_{ref} + ND_{lost}$, as shown in Figure 4.3. By doing so, the network is adaptive to unexpected nodes failures. Intuitively, when a node detects neighbor failures, it is supposed to raise the expected number of neighbors, so that it has more opportunity to get a path to other nodes. This is very useful in a large and distributed network, like the IoT. It is desirable to design localized or distributed algorithms/protocols that can adapt to the network change, which is the main goal of this work.

The rationale of using probability model here and in the training process are the following: it might work very well if get rid of the probability model, because the close-loop feedback can adjust the CR to approximate ND to ND_{ref}, but the probability model here can be viewed as the prior knowledge that is useful to make the control system act on the changes faster and more accurate. In fact, the probability model is more suitable for large network, but it may not work well if the network is very small, say, tens of nodes. The probability model is useful for the case that the network is large, and it worth to investigate from the theoretical study perspective. But, for a small network, the probability model may fail as there are not enough samples/nodes. Considering this may happen in reality and to complete this research, another approach will be proposed in this work (see Figure 5.3), and will be evaluated by real sensor nodes. For this new control system, there is no probability model.

4.2.1 LFTC learning data set

Provided a training data set from equation (4.2), the fuzzy controller can be obtained through learning algorithms, such as neuro-adaptive learning algorithm. With the help of the Matlab "adaptive neuro-fuzzy inference system" (ANFIS) tool, FTC controller is not difficult to get. In ANFIS, user has to provide the basic shapes of the membership function, e.g. Triangular, Trapezoidal, Guassian etc. Then, the membership function parameters are automatically tuned through a BP algorithm individually or in combination with the least square method. Once the data training is completed, the fuzzy inference system is available by simply using the function "evalfis" provided by Matlab. There are some other open source libraries to implement the BP algorithm, though.

The machine learning algorithms, in general, can be used here as a predictor, with or without fuzzy-logic. The reason here using the neural network is to learn the parameters in the fuzzy-logic system automatically. Other advance machine learning algorithms maybe work as well, such as logistic regression, linear regression, etc. In particular, on-line machine learning algorithm can play an important rule in the self-adaptive network, as it can learn and adjust the parameters at runtime. Depends on the complexity of the on-line learning algorithms, they may or may not be able to help improve the network performance. Applying machine learning algorithm to solve the problem, however, is out of the scope of this work.

Here, this work concentrates on how to generate the training data set. As illustrated in Figure 4.2 and equation (4.2), the inputs are ND_{ref} and $Prob$, and the output is CR. The training data has to be in the form that all training data group together with the last column being the outputs. Given ρ, $ND_{ref} \in \{k_1, k_2, \cdots k_m\}$ and $CR \in \{r_1, r_2, \cdots, r_j\}$, $Prob = f(ND_{ref}, CR)$ can be calculated from equation (4.2). The training data set $\boldsymbol{T}$ is a $s \times 3$ matrix in the form of

$$[ND_{ref}, Prob, CR]$$

where $s = m \cdot j$. For instance, one item in the training data set could be [3, 0.9, 25], which can be interpreted as CR is set to 25m if the probability that $ND_{ref} \geq 3$ is 0.9.

Despite the equation (4.2) is the only one used to generate training data, as the networks are assumed to be randomly and uniformly distributed. Other ND distribution, however, in real network exist, such as non-Poisson equation (4.3) [59].

$$P_2(ND \geq k) = f_2(r, k) = 1 - \sum_{N=0}^{k-1} \binom{n-1}{N} p(r)^N (1 - p(r))^{n-1-N}. \tag{4.3}$$

4.2.2 LFTC algorithm complexity

The key component of the training algorithm is neural network, of which algorithm complexity depends on the size of training data set, but the upper bound of the algorithm (both space and time) complexity is unknown, because it depends on the structure of the neural network chosen. It is unrealistic to implement the training process in the resource limited devices, e.g. sensor nodes. However, it only has to perform the training process once for a specific deployment, so in practice the fuzzy-logic can be obtained from a powerful machine and import the fuzzy-logic inference system to resource limited nodes later. Taking the Matlab simulation tool as an example, it is able to training and validating the fuzzy-logic inference engine, and then generate a complete fuzzy-logic engine as a whole to run it on a node. The training process is like a black-box for users.

The de-facto JAVA fuzzy-logic implementation, "jFuzzyLogic", is still too heavy for most sensor nodes in current market. An alternative is to run it in more powerful devices, and only distribute the outputs of fuzzy-logic controller to sensor nodes. But it somehow breaks the network structure if the network intends to be completely distributed, because it has to be centralized structure to do this. In a small network, centralized approach will work, hence the fuzzy-logic controller can run in powerful nodes, e.g. cluster-head or base station.

4.2.3 Controller parameters

The network is stochastic, so the ND must be uncertain as well. The fuzzy-logic controller merely provides the quantities of CR that is very likely to have the desired ND, but it can not 100% guarantee that it will obtain the desired $ND = ND_{ref}$. Therefore, some conventional approaches are needed to further fine-tune the network. For instance, if $CR = 25\,\mathrm{m}$ can not lead to actual $ND = ND_{ref}$, then next step is to adjust $Prob$ to a higher value according to the node degree error e_{ND}. As shown in Figure 4.2, there is an integral controller outside the fuzzy control to adaptively change $Prob$. From the control theory point of view, the system properties (e.g., steady state) are controlled by parameter $Prob_0$ and K. If e_{ND} is less than 0, K is configured to be half of its initial value. As a result, CR tends to increase faster than decrease.

Unlike other ways to analyze the parameters in the control system, due to the complexity of the system, the explicit transform functions are unknown. Instead, this work applies the numerical approach to evaluate the performance of the control system. In general, the parameters in the control system can guarantee that the whole system converges, but for individual node in a network, it may not necessarily converge. Next section will give more details. This also brings another important issue: the *synchronization* problem. Nevertheless, the intuition is that the close-loop control is able to make the node degree to be the desired one, meanwhile the whole network also achieves the desired node degree. More importantly, if the time interval that updates CR to be long enough can be controlled, then the whole network is approximately synchronized. So, this work did not consider the synchronization problem. Rather, each node changes its CR at the same time in the simulations, and updates its CR in a asynchronous manner when it comes to real implementation.

In Section 4.5, the impact of $Prob_0$ and K will be further discussed by simulation.

4.2.4 LFTC protocol

According to the design of LFTC in this section, corresponding LFTC protocol is presented. Each node broadcasts the HELLO message at the maximum communication range r_{max}, in order to collect neighbors' information. The communication range is modified accordingly based on the fuzzy-logic controller.

On the one hand, it is only considered the undirected links (that is, there is a link between two nodes if and only if they are both in each other's communication range), because in practice many routing protocols assume that the link between two nodes are undirected by default. On the other hand, LFTC is a localized control algorithm running at each node independently. As a result, the node degree changes over time. For instance, node u is the neighbor of v at this round of communication range updating, but it is likely that in next round u is not the neighbor of v due to the communication range of u and/or v being altered. Section 4.5 will show that the system goes to steady state by appropriately defining the value of "rounds" in the LFTC protocol. Next, the LFTC protocol is presented:

Algorithm 4.2.1. *LFTC protocol (for a generic node i)*

```
Require:
1: Training data set, $\boldsymbol{T} = [\boldsymbol{k}, \boldsymbol{Prob}, \boldsymbol{CR}]$;
2: Maximum communication range, $r_{max}$;
3: Reference node degree, $ND_{ref}$;
4: Initial Probability, $Prob_0$;
5: Initial $K$, $K_0$;
6: Number of rounds, $rounds$;
Ensure:
7: Obtain the fuzzy-logic control system by ANFIS, ANFIS($\boldsymbol{T}$);
8: $CR_i \Leftarrow r_{max}$;
9: $Prob \Leftarrow Prob_0$;
10: while $rounds > 0$ do
11:     Broadcast HELLO message with current $CR_i$;
12:     For messages received from other nodes, store the ID of its neighbors;
13:     Calculate the number of neighbors $ND$ in the neighbor list;
14:     Calculate $e_{ND} = ND - ND_{ref}$;
```

```
15:     if $e_{ND} < 0$ then
16:         $K = K_0$;
17:     else
18:         $K = \frac{K_0}{2}$;
19:     end if
20:     $Prob \Leftarrow Prob - K \cdot e_{ND}$;
21:     $CR_i \Leftarrow evalfis(ND_{ref}, Prob)$; %The output $CR_i$ can be calculated by $CR_i = evalfis(k, Prob)$
22:     $rounds \Leftarrow rounds - 1$
23: end while
```

4.3 Fuzzy-logic control based on if-then rules

Unlike the controller in Section 4.2, the alternative is to create the controller based on rules.

◇ *Rules-based Fuzzy-logic Topology Control System Design*

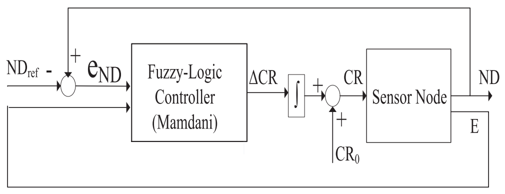
In this section, another fuzzy logic controller is proposed. It is called Rules-based Fuzzy-logic Topology Control (RFTC) because of the need to design the if-then rules. Unlike LFTC, RFTC is depicted in Figure 4.4(a). Here, the fuzzy controller of RFTC is not automatically generated from the training data set, and the fuzzy rules and membership functions are generated by the heuristic approaches or experiences instead. The input parameters are different as well. The input variable probability $Prob$ is replaced by residual energy E, and e_{ND} becomes input. The output is the CR incremental, ΔCR. CR_0 is the initial value of a sensor node, which is random for each node. This thesis leverages "Mamdani" type fuzzy inference system.

◇ *Membership Functions and If-then Rules*

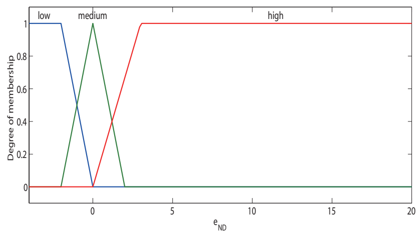
For each input and output, there are three fuzzy sets: High, Medium, and Low. Their membership functions are shown in Figure 4.4(b), 4.4(c), 4.4(d). Intuitively, if e_{ND} is High and E is High, ΔCR should be Low; if e_{ND} is Low, no matter what E is, ΔCR should be High, because maintaining the network connectivity is the top priority. The details of if-then rules are shown in Table 4.1.

The design of membership functions and if-then rules is rather heuristic. The change of membership functions and if-then rules has significant impact on the performance. It is necessary to tune the membership shapes and positions, and change the rules according to different network deployment strategies and network models.

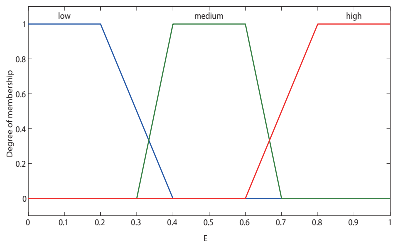
By defining if-then rules and membership functions, a complete fuzzy-logic is finished. The design of the fuzzy-logic likes this is usually not a difficult task, while the parameter tuning is. As mentioned in Section 4.1, fuzzy-logic is like a bridge between domain experts and engineers/programmers. It is quick to start the design and get the



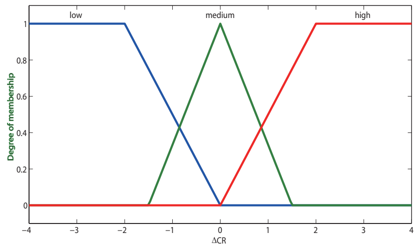
(a) Fuzzy-logic Topology Control



(b) Membership Function of e_{ND}



(c) Membership Function of E



(d) Membership Function of ΔCR

Figure 4.4: *Rules-based Fuzzy-logic Topology Control (RFTC).*

Table 4.1: *Fuzzy-Logic If-Then Rules.*

e_{ND} \ E	High	Medium	Low
High	Low	Medium	Medium
Medium	Medium	Medium	Low
Low	High	High	High

prototype running. The rest of the time will spend on the parameter tuning.

4.4 Matlab-based simulation tool

There are lots of tools (e.g. NS2, NS3, OPNET, OMNET++, Matlab, etc. [137]) to simulate various kinds of networks, such as Ethernet, WiMax, 3G, 4G, WiFi, ZigBee, etc. The choice of simulator is depended on the preference of researchers and specific problems that are going to study. There is no such a tool that is good for all networks and problems.

Among those tools, Matlab [100] is an excellent and popular one in the research community. This research intended to investigate the control algorithms (in particular, the fuzzy logic control), and parameter optimization of controllers. On the one hand, Matlab provides the mature model and tool for developing and validating fuzzy logic controller. On the other hand, a complete network implementation might too heavy for

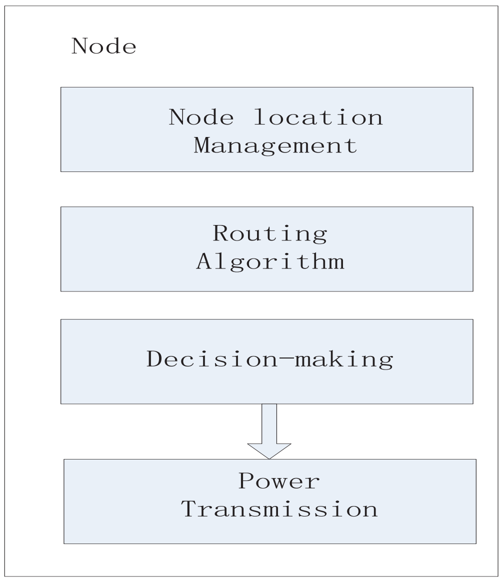


Figure 4.5: *Simplified node model in Matlab-based simulator.*

the optimization because of its iterative nature, e.g. maybe need thousands of iterations. If only the necessary models are implemented, a lightweight simulator is needed in order to run much faster. Matlab, therefore, becomes the first choice of this thesis.

Matlab, however, to some extend is not the best simulator for wireless sensor network simulation due to the lack of complete layered network stacks that compliant with various wireless standards. Part of this work is to develop a dedicated network simulator based on Matlab environment, which directly models the wireless network features that important to the problems of this thesis, and ignore irrelevant functionalities. In fact, since the focus of this work is the network connectivity and power related problems, the most important functionalities are finally implemented in the progress of developing this thesis: node location management model, power transmission model, routing algorithm, decision-making logic; besides, the network visualization model and the outcome analysis/display models are developed to better present the results. Figure 4.5 illustrates the main function blocks that are implemented in the simulator. The decision-making logic is the place where the controllers are realized.

In this chapter, and the following chapters of this thesis whenever needs a simulator, the analysis is based on this Matlab-based simulator.

4.5 System validation

The controllers proposed are validated by using the Matlab-based simulator. This section will first try to solve the problem of choosing control parameters, then present

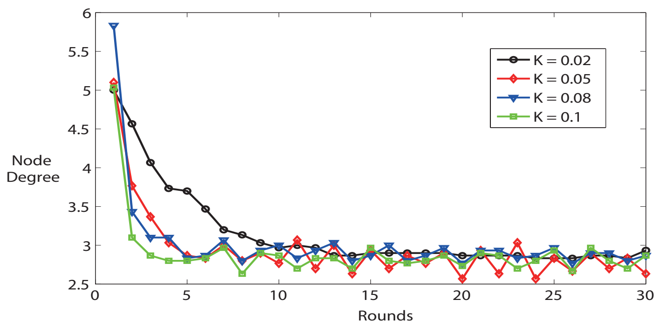


Figure 4.6: *Average node degree with different K ($Prob_0 = 0.8$, n = 60, k = 3).*

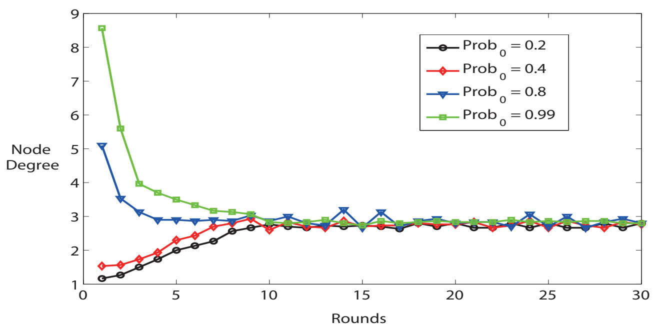


Figure 4.7: *Average node degree with different $Prob_0$ (K = 0.02, n = 60, k = 3).*

the network models that used in the simulations. Comparisons are made with other algorithms to show the fault-tolerant improvement of the proposals. Therefore, Section 4.5.2 will briefly introduce these algorithms. For more details, readers can consult the provided references.

4.5.1 Control system analysis

As mentioned in Section 4.2, FTC is affected by two parameters ($Prob_0$ and K) outside the fuzzy-logic controller. The simulator provides average node degree calculated by the sum of node degree for all nodes divided by the number of nodes in the network. Figure 4.6 shows the impact of K on network settling time, accuracy and stability. Firstly, the average node degree quickly converges to its steady state value (after about 10 rounds). Secondly, the average node degree converges to the reference node degree $k = 3$. A lower K results in a more stable system, but longer convergence time. For instance, $K = 0.02$ is more stable than $K = 0.1$, but $K = 0.02$ takes longer time to reach steady state. Thirdly, the system is stable, but the system demonstrates heavier oscillation behavior when K is bigger.

Figure 4.7 illustrates different features at the first several rounds. If $Prob_0$ is higher at the beginning, the nodes are more likely to be connected than $Prob_0$ is lower. Nevertheless, the average node degree goes to stable state after 10 rounds as well. As a result, $Prob_0$ has an impact on the initial status of network. In short, the simulation results illustrate that K has an important influence on network settling time, accuracy and stability. In the following simulations, $K = 0.02$, $Prob_0 = 0.8$ and $rounds = 15$ are selected to configure the parameters described in FTC protocol.

4.5.2 Representative algorithms

There are many topology control algorithms/protocols proposed in the literature. Some of the state-of-the-art topology control algorithms are: Local Tree-based Reliable Topology (LTRT) [102], Local Minimum Spanning Tree (LMST) [89] and Fault-tolerant Local Spanning Subgraph (FLSS) [88], which are similar to each other as they are based on the spanning tree algorithm. This thesis chooses LTRT as the representative algorithm in this category. On the other hand, the List-based topology control [35] is selected to represent the algorithm that does not rely on constructing the spanning tree but utilizes the neighbors' information. In addition, this work also compares them with the one without any control approaches or algorithms, which means that the CR is not changed during the simulation. It is called NONE algorithm in this thesis.

◇ ***NONE***

Once all sensor nodes are deployed in the field, each node configures its communication range to the maximum communication range, and the communication range does not change during the simulation. NONE generates the most connected network, thus gives the upper bound of network connectivity. NONE algorithm is used to simulate the case that there is no topology control applied to the IoT.

◇ ***LTRT: Local Tree-based Reliable Topology [102]***

Local Tree-based Reliable Topology (LTRT) is a localized algorithm. Basically, it is a variant of spanning tree algorithm. When conducting spanning tree algorithm k times, the resultant network is a k-edge-connected if the original network is at least k-edge-connected network. More specifically, it repeatedly processes the network k times as follows: given a s-edge-connected network $G(V, E)$, where V is the set of nodes, E is the set of links, and $s \geq k$. First one of its spanning tree $T(V, \hat{E}_1)$ is calculated by a localized algorithm, then all links in $\hat{E}_1$ from E are removed and the resulting network is denoted as $G(V, E - \hat{E}_1)$. Next time, the same process is performed on $G(V, E - \hat{E}_1)$. Repeating this process k times, the resultant network will be $G(V, E - \hat{E}_1 - \hat{E}_2 - \cdots - \hat{E}_k)$. The final k-edge-connectivity network is formed by combining all trees together, that is $G(V, \hat{E}_1 + \hat{E}_2 + \cdots + \hat{E}_k)$. The final CR of each node is selected from the maximum CR that connects to its neighbors in $G(V, \hat{E}_1 + \hat{E}_2 + \cdots + \hat{E}_k)$.

The LTRT requires that the original network is at least a k-edge-connected network, and it requires the location information of its neighbors. LTRT needs that each node runs at its maximum CR before starts running the algorithm. LTRT has been compared with Cone-Based distributed Topology Control CBTC(α) [87] and Fault-tolerant Local Spanning Subgraph ($FLSS_k$) [88]. $FLSS_k$ is a near optimal algorithm with high complexity. The simulation results of LTRT show that LTRT achieves comparable performance as that of $FLSS_k$, but at a much lower cost.

◇ ***List-based Topology Control [35]***

List-based topology control is a cooperative algorithm. It is called list-based because the change of CR relies on the list of its neighbors. Each node does not change its CR (increases or decreases) until its neighbors requires its CR to be changed. In other words, each node is able to ask for its neighbors to change their CR when it needs them to do so. If a node has more neighbors than it needs, it will request the closer neighbors to change their CR, and other neighbors will remain their CR. For instance, if node u wants its ND to be 4, it broadcasts a request message. All nodes within its CR will receive the request, and they change their CR to reach node u. If there are more than 4 nodes that can reach u, only 4 nodes closer to u finally increase their CR, other nodes will not modify their CR.

◇ ***Fault-tolerant Local Spanning Subgraph (FLSS)*** [88]

Each node runs at their maximum communication range to collect neighbors' information, e.g. IDs, locations. Based on this information, each node first sorts all edges in an ascending order, and then only selects edges according to the order if it is not $k-connected$ to it. Maybe some edges are directed, but it is optional to only consider bi-directional edges by deleting directed edges, or turning directed edges into bi-directional edges.

◇ ***k-Neighbor algorithm*** [22]

Each node first runs at its maximum communication range as well, in order to collect neighbor IDs and location information. Its final communication range is set to the distance between itself and its $k-th$ nearest neighbor. One of the important features of k-Neighbor is that the algorithm is asynchronous. Each node starts running its topology algorithm at a random time. However, the difference between nodes wake up is bounded by a known constant Δ. By the time $4\Delta + 2d + \tau$, all nodes terminate setting the communication range, where d is the time to obtain a specific contention free wireless with probability P, and τ is the upper bound of processing time for sorting neighbors. In the circumstances that the number of neighbors is less than k, k-Neighbor algorithm modifies its communication range to the distance to the farthest neighbor.

The List-based topology control is a localized algorithm, but it needs the location information of its neighbors as well, because the length of CR needed is calculated according to their location information.

4.5.3 Network model and configurations

Before starting the simulation, the network model and configurations are introduced.

- Training data set can be obtained according to different network models. The disk model is employed, which means that CR is modeled as a disk with radius r. A link exists between two distinct nodes only when they are both in each other's CR, thus all links are undirected. All nodes are randomly and uniformly deployed in a $100 \times 100m^2$ field, therefore only equation (4.2), rather than equation (4.3), is used in the fuzzy-logic leaning process.
- All nodes in the field are stationary after the deployment.
- Each node is capable of adjusting its CR between [10,30]m. In addition, the initial CR of each node is a random value chosen from [10, 30]m in order to simulate heterogeneous IoT in terms of CR.
- There is a special node in the network called base station (BS) located at the center of the field.
- Each node transmits sensor data to BS periodically. Each node updates its CR according to different control approaches or algorithms after transmitting 800000 bits packets. It is called one "round" simulation. Note that 800000 bits packets are not necessary to be transmitted at one time. They can be fragmented into many small packets.
- The routing algorithm is the shortest distance to BS. The simulation is terminated when BS no longer receives packets.
- The energy dissipation model is the same as [11]. Equation (4.4) and (4.5) represent power consumed when a node transmits/receives a L bits packet to/from another node at distance d. Constant $E_{elec} = 50\,nJ/bit$ and $\epsilon_{amp} = 100\,pJ/bit/m^2$ are related to the circuit and antenna design of sensor nodes. Each node is charged with 1J energy at the beginning of simulation. Nodes stop sending or receiving packets when there is no battery left.

$$E_{tx} = L \times E_{elec} + L \times \epsilon_{amp} \times d^2. \tag{4.4}$$

$$E_{rcv} = L \times E_{elec}. \tag{4.5}$$

- Apart from running out of battery, in order to simulate random attacks or damage by malicious people or nodes, a configurable parameter called failure probability is introduced in the simulation. Each node experiences identical failure probability at each round.

4.6 Simulation results and analysis

This section presents the simulated results and analyzes the outcomes. Moreover, this section evaluates how the ND will be controlled by the two controllers, and whether the control goal is attained in the absence and in the presence of random failures. The energy-related problems also are discussed.

4.6.1 Resultant topology

LFTC is compared with other algorithms. Figure 4.8 shows an instance of the topology after running different topology algorithms on the same network when $k = 3$ and $n = 65$. Figure 4.8a indicates that the network is very well connected, because each node runs at the maximum communication range, but it also causes severe signal inferences in the network, as well as too much energy consumption. LTRT in Figure 4.8d shows the second most connected network. FLSS in Figure 4.8c, k-Neighbor in Figure 4.8b and FTC in Figure 4.8e are quite similar. They preserve network connectivity with much lower amount of links between nodes, which means that much less signal inferences are introduced. Figure 4.8 implies that FLSS, k-Neighbor and FTC demonstrate the improvement in terms of the tradeoff between network connectivity and link quality. FLSS may produce directional links. The undirected links are plotted, so the resulting network running FLSS shown in Figure 4.8c is not a $3-connected$ network, the original network shown in Figure 4.8a is at least a $3-connected$ network though.

4.6.2 Average node degree

Simulations are divided into two parts. In the first part, it is only considered the effect of the topology control approaches or algorithms on the initial network topology. In other words, it is only observed that the topology changes after all nodes are deployed without any data transmission in the network. Therefore, some configurations in Section 4.5.3 are not applicable to this simulation part, such as the energy dissipation model, the routing algorithm, and the failure model. The second part simulates the networks with packets being sent at each node. The main differences between two parts are: in the second simulation part, the energy of each node will be decreasing and some of the nodes may run out of battery during the simulations. Particularly, the relay nodes deplete energy faster. As a result, the links between nodes are dynamic. Because E is one of the inputs for RFTC, the energy status has an impact on the controller output. Besides, the failure probability also influences the connections among nodes. Therefore, the second part simulation is a more dynamic scenario.

For each part, the deployment area is fixed but the number of nodes deployed varies from 30 to 75 nodes to change the node density. Since the deployment is random, 50 different networks are generated for every algorithm with the same configurations (e.g.

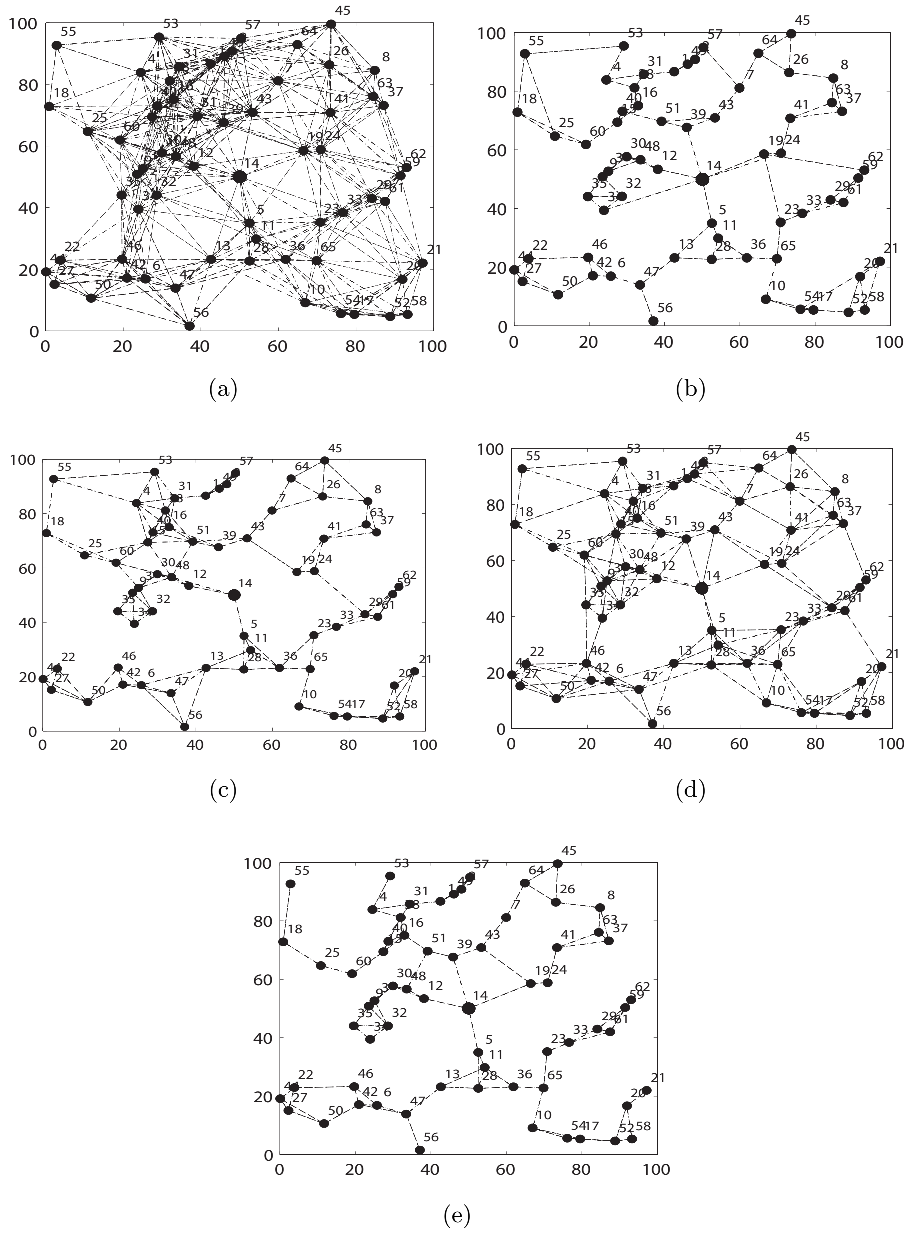


Figure 4.8: *Example of resultant topology (n = 65, k = 3). (a) NONE; (b) k-Neighbor; (c) FLSS; (d) LTRT; (e) FTC.*

same number of nodes and failure probability). Obtained results are the average of 50 networks.

4.6.3 Topology control on initial network

Figures 4.9(a), 4.9(b), and 4.9(c) show the average node degree, which is calculated by the sum of the node degree of all nodes divided by the number of nodes in a network. As mentioned in Section 4.5.3, the link between two nodes is undirected. A node connected by a directed link is not counted as a neighbor. It is observed that two approaches proposed, LFTC and RFTC, are able to trace the reference $k = 2, 3, 4$ as the number of nodes deployed in the field increases. But Figure 4.9(d) shows that the network is unable to trace $k = 5$ when the node number is less than 60. Because the maximum CR is limited, it is less likely that each node has node degree at least 5 if the network density is not high enough. On the contrary, other algorithms are unable to trace the desired k. LTRT has the highest ND, because it is the most aggressive algorithm, which needs each node runs at maximum CR before starts running LTRT. Higher average ND is good for the network connectivity, but also introduces higher signal inferences.

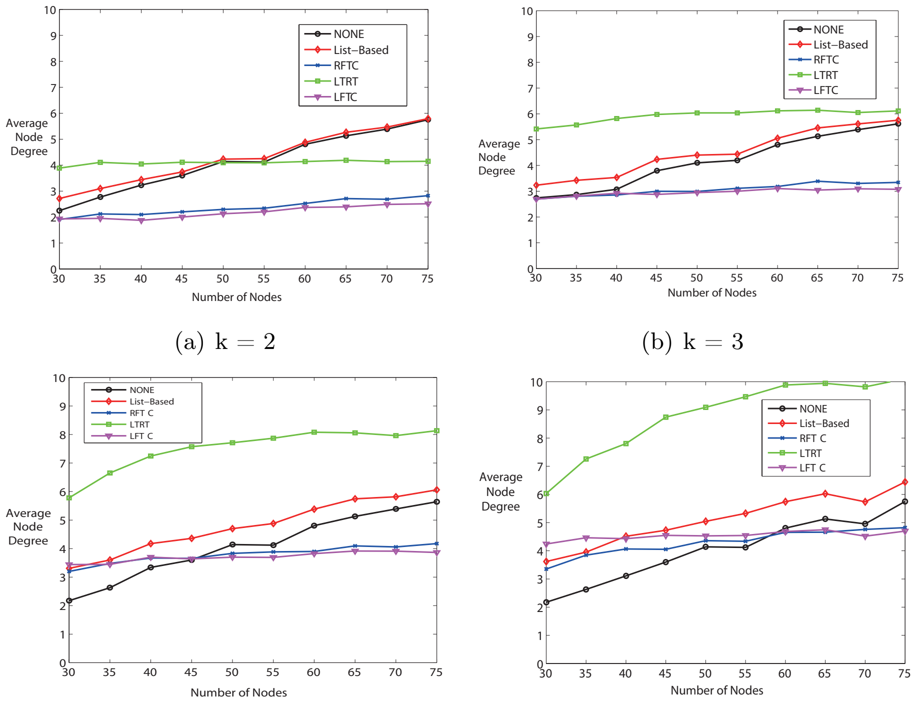
It is worth noting that LFTC and RFTC are very close to each other in Figures 4.9(a), 4.9(b), and 4.9(c), but in Figure 4.9(d) LFTC demonstrates improved performance at tracing the desired k than RFTC when the nodes deployed are less than 60. LFTC manifests improved adaptivity than RFTC in response to the network dynamics. As mentioned in Section 4.3, RFTC has to tune the control parameters to adapt the network dynamics, but the control parameters are the same for RFTC in the simulations.

As shown in equation (4.4), energy consumption is propositional to the squared distance between a transmitter and a receiver. Similar to the way to evaluate LTRT [102], this work uses average CR (CR_{avg}), average maximum CR (CR_{max}), and Energy Expended Ratio (EER) (defined as $EER = 100 \times \frac{CR_{avg}}{CR_{max}}\%$) to evaluate network energy consumption. Figures 4.10, 4.11, and 4.12 illustrate the average CR and average maximum CR, and its EER when $k = 3$, respectively. The CR_{avg} of LFTC, RFTC, and LTRT decreases as the number of nodes increases, but the network running List-base algorithm and NONE, the CR_{avg} does not change to a great extent. They are expected results because:

(1) for LFTC, RFTC, and LTRT, if there are more nodes deployed, it indicates that the node density becomes higher. Therefore, the lower CR can obtain desired k neighbors. In other words, energy consumption is higher when the network is sparse, while energy consumption is lower when the network is dense.

(2) for the algorithm NONE, because the CR does not change and the initial CR randomly choose from [10,30]m, therefore its CR_{avg} is always about 20m.

(3) for the List-base algorithm, a node can ask for other nodes, which are not its neighbors but within its CR, to increase their CR, but the maximum CR between any



(a) k = 2

(b) k = 3

(c) k = 4

(d) k = 5

Figure 4.9: *Average Node Degree with Different k.*

two nodes is limited by maximum CR between them. For instance, nodes u and v have communication ranges of 15m and 20m, respectively. The node v may need u to increase CR to reach v, however the maximum between u and v is impossible bigger than maximum between them, that is 20m. In other words, the node CR can increase but the incremental is limited.

As far as the average energy is concerned, this thesis concludes that the energy consumption of LFTC and RFTC are always lower than LTRT regardless of the node density, and LFTC and RFTC outperform List-based and NONE after the number of nodes higher than 40. LFTC improved performance little bit than RFTC. As far as the

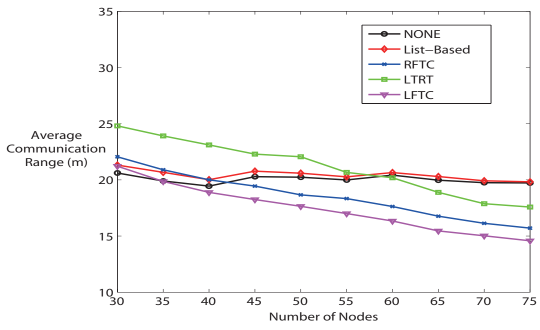


Figure 4.10: *Average Communication Range (k = 3).*

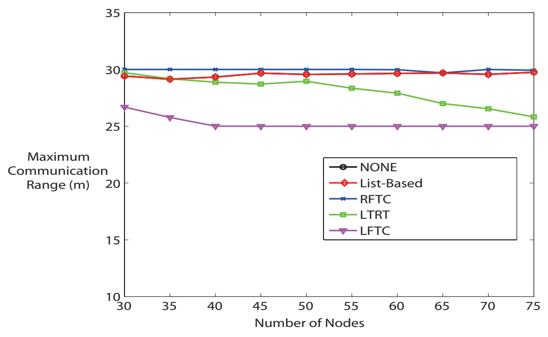


Figure 4.11: *Maximum Communication Range (k = 3).*

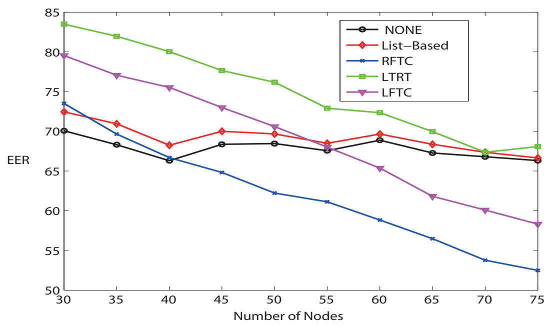


Figure 4.12: *Energy Expended Ratio (EER) (k = 3).*

most power consumption node is taken into account, as shown in Figure 4.11, LFTC has the lowest maximum CR. It indicates that the most power consuming node running LFTC in a network has the lowest power consumption than running other algorithms. EER in Figure 4.12 shows the same trend as Figure 4.10. It is expected that EER is low. RFTC maintains lowest EER than other algorithms. LFTC is higher than RFTC due to the maximums being lower than RFTC, as shown in Figure 4.11.

In short, the simulations performed in this section only focus on the network connectivity and corresponding energy consumption. The simulation results imply that two proposals (LFTC and RFTC) are able to maintain the desired node degree, which show the effectiveness of the feedback control loops, while the resulting node degree are higher than expected when using other conventional methods. On the other hand, from the energy consumption point of view, the proposals manifest lower energy consumption on the resulting networks than other algorithms.

4.6.4 Topology control on network with random failures

This section evaluates the network performance when the nodes send and receive sensor data periodically. Moreover, this work considers not only nodes running out of battery, but also nodes that are damaged on purpose. The node damage is modeled by introducing random failures at each round. In the simulation, there are 60 nodes

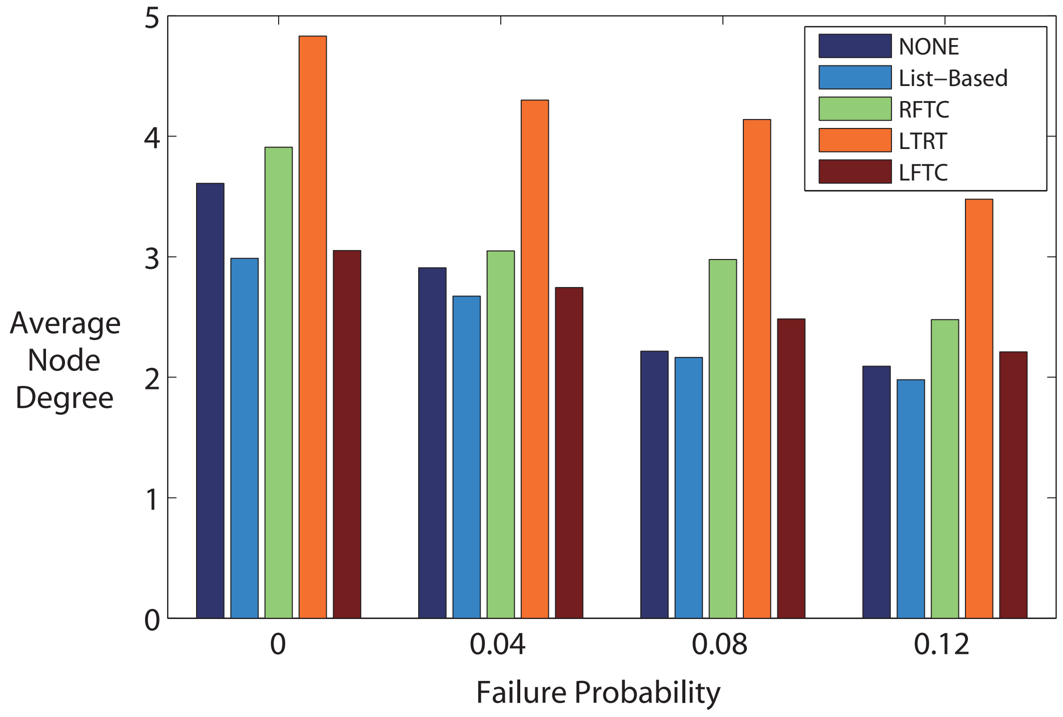


Figure 4.13: *Average Node Degree for Different Failure Probabilities (k = 3, n = 60).*

deployed in the field. Since each node leaves the network randomly at each round, ND_{avg} is calculated in a different way. For any node i in the network, let ND at round j is denoted as $ND(i, j)$, and $round(i)$ be the number of rounds before node i runs out of battery or is attacked. Average ND of node i, denoted as $ND_{avg}(i)$, is calculated by the sum of ND at each round divided by the number of rounds for node i, as shown in equation (4.6). The ND_{avg} of a specific network is the average of $ND_{avg}(i)$ for all nodes, as shown in equation (4.7), where n is the total number of nodes deployed.

$$ND_{avg}(i) = \frac{\sum_{j=1}^{round(i)} ND(i, j)}{round(i)}. \tag{4.6}$$

$$ND_{avg} = \frac{\sum_{i=1}^{n} ND_{avg}(i)}{n}. \tag{4.7}$$

Figure 4.13 shows the ND_{avg} with failure probabilities 0%, 4%, 8%, and 12%. First of all, ND_{avg} decreases for all algorithms, because the failure probability increases. Second, ND_{avg} of LFTC and RFTC decreases slower than others (especially when failure probabilities are 4% and 8%), implying that the proposed approaches are able to effectively resist ND_{avg} (or, network connectivity) decreasing as the failure probability increases. RFTC outperforms LFTC, but both worse than LTRT. Third, ND_{avg} of List-base and NONE are very close, while LTRT still has the highest ND_{avg}.

In this section, the simulations are performed on the more realistic scenarios, where nodes transmitting, receiving, and routing packets, meanwhile considering that nodes are experiencing running out of battery and malicious attacks. The results indicate that the control approaches presented are still valid. The performance degradation is slower than other algorithms in the case that random failures occur.

4.7 Summary

In order to improve the network connectivity in the presence of node failures, this chapter proposed two localized and energy-efficient approaches, called LFTC and RFTC, based on the fuzzy-logic. As for LFTC, the fuzzy-logic controller is obtained through the training data set, while the fuzzy controller is based on the heuristic if-then rules and membership functions for RFTC. However, both approaches can achieve almost the same goals. LFTC and RFTC both have strengths and weaknesses. The main benefit of LFTC is that it relies on a mathematical model, so there is no need to adjust fuzzy controller parameters, but the mathematical model may be not available or accurate enough for realistic sensor nodes deployment. In contrast, the control parameters of RFTC do not depend on the mathematical model, but the parameters have to be tuned according to a specific node deployment to achieve the best performance. It could be not an easy task, or even not feasible.

Compared LFTC and RFTC with other three algorithms NONE, LTRT, and List-based by simulations, two proposals of this thesis can achieve desired node degree and save more energy. Furthermore, in the case that random node failures exist, such as nodes running out of battery or suffer intentional attacks, LFTC and RFTC show their capability to resist node failures by adaptively adjusting the communication range. Additionally, the simulations employ the disk model which is an ideal wireless channel. Nonetheless, the approaches of this work can be extended to more realistic models as long as the node degree is known, e.g. binomial distribution [59]. The difference is the way to obtain a training data set.

Considering that the control system created upon a probability model requires large scale deployment, so it is evaluated by computer simulation. These results have been published in [65][64]. What has implemented in real sensor nodes, however, is another control system with two-loop which will be explored in Chapter 5.

Chapter 5

A Lightweight Self-adaptive Control System Design and Optimization

The controllers discussed in Chapter 4 have their advantages that can deal with large scale probabilistic network, and they are distributed. This chapter starts with a new proposal with a control system that consists of two loops. It still uses the fuzzy logic as its core reasoner. Based upon the two-loop controller, a lightweight but more applicable controller is proposed, which will be eventually implemented in a real application (see Chapter 6). Furthermore, another important research which will be involved in this chapter is that this thesis applies different kinds of optimization algorithms to optimize the controller in order to further improve the overall performance.

5.1 Two-loop controller

An alternative to face the network connectivity and energy consumption issue is to design a control system based on single feedback loops for self-adaptation of communication range against dynamic changes in the links with the neighboring nodes. A basic feedback control loop should contain a function of decision-making (FDM) which provides a variation factor of the communication range (Δcr) as output and the node degree error (e_{ND}) as input (i.e. the difference between the desired value ND_R and the real one ND).

Additionally, another single feedback loop can be added to perform the task of modifying the desired value for the node degree according to the battery level (E). The aim of this action is to reduce the device power consumption when the battery is under

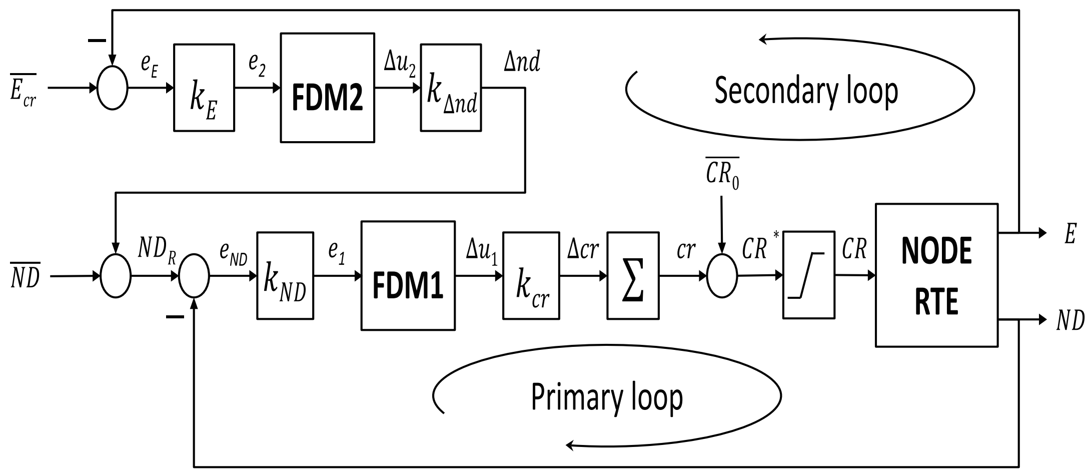


Figure 5.1: *Reconfiguration loops for a self-adaptive communication range.*

a certain energy critical level ($\overline{E_{cr}}$). A reduction of the node degree means to reduce the device communication range by reducing the signal transmission power. Hence, this represents a reduction of the energy consumption and an increment in the battery lifetime. The controller (FDM hereinafter) of this loop receives the difference between a battery critical level ($\overline{E_{cr}}$) value and the actual battery level (E) as input.

Figure 5.1 illustrates both reconfiguration closed loops with two FDM strategies, a "primary" loop for the calculation of Δcr (**FDM1**) and "secondary" loop to determine the desired or reference value of the node degree (**FDM2**).

Within the **primary reconfiguration loop**, the communication range can be updated at certain time instant k as follow:

$$e_{ND}(k) = ND_R(k) - ND(k). \tag{5.1}$$

$$e_1(k) = k_{ND} e_{ND}(k), \quad \Delta cr(k) = k_{cr} \Delta u_1(k). \tag{5.2}$$

$$\Delta u_1(k) = f_{DM1}\big(e_1(k)\big). \tag{5.3}$$

$$\begin{gathered} CR^*(k) = cr(k) + \overline{CR_0}. \\ cr(k) = cr(k-1) + \Delta cr(k). \end{gathered} \tag{5.4}$$

$$CR(k) = \begin{cases} CR_{min}, & CR^*(k) < CR_{min} \\ CR^*(k), & CR_{min} \leq CR^*(k) \leq CR_{min} \\ CR_{max}, & CR^*(k) > CR_{min} \end{cases} \tag{5.5}$$

At the above equations, $\overline{CR_0}$ is the initial value of the communication range and f_{DM1} represents the **FDM1** function. The function input e_1 is the normalized node degree error and the output Δu_1 is the normalized communication range variation factor. Also, the value of CR is saturated between its minimum value CR_{min} and maximum value CR_{max}. k_{ND} and k_{cr} are normalization or scale factors for the input and the output respectively. Both factors can be calculated as:

$$k_{ND} = 1/\overline{ND}, \quad k_{cr} = \overline{\Delta cr}, \tag{5.6}$$

where $\overline{ND}$ represents the nominal or desired value of the node degree when the battery has a critical energy level ($\overline{E_{cr}}$). Factor $\overline{\Delta cr}$ is the communication range variation rate.

The **secondary loop** tunes the desired value for the node degree according to the battery level and can be formalized as follow:

$$e_E(k) = \overline{E_{cr}} - E(k). \tag{5.7}$$

$$e_2(k) = k_E e_E(k), \quad \Delta nd(k) = k_{\Delta nd} \Delta u_2(k). \tag{5.8}$$

$$\Delta u_2(k) = f_{DM2}\big(e_2(k)\big). \tag{5.9}$$

$$ND_R(k) = \overline{ND} + \Delta nd(k). \tag{5.10}$$

where e_E is the difference between the battery critical level and the actual level and Δnd is the node degree variation factor. Besides, f_{DM2} represents the **FDM2** function where its input e_2 is the normalized or scaled value of e_E and output Δu_2 is the scaled node degree variation factor. k_E and $k_{\Delta nd}$ are scale factors for the input and the output respectively. Both factors can be calculated as:

$$k_E = 1/\overline{E_{cr}}, \quad k_{\Delta nd} = \overline{\Delta nd}, \tag{5.11}$$

where $\overline{\Delta nd}$ is the node degree variation rate.

In general terms, both loops can be seen as a function which receives the actual values of ND and E as inputs, returning a new value of CR as output and requiring a set of parameters $\overline{\mathbf{P}}$:

$$\begin{gathered} CR(k) = g\big(ND(k), E(k), \overline{\mathbf{P}}\big). \\ \overline{\mathbf{P}} = \Big[\overline{CR_0}, \overline{ND}, \overline{E_{cr}}, \overline{\Delta cr}, \overline{\Delta nd}, CR_{min}, CR_{max}\Big]. \end{gathered} \tag{5.12}$$

where the complete parameters list is: initial value of the communication range ($\overline{CR_0}$); desired value of the node degree when the battery has a critical energy level ($\overline{ND}$); critical energy level ($\overline{E_{cr}}$); communication range variation rate ($\overline{\Delta cr}$); node degree variation rate ($\overline{\Delta nd}$); minimum and maximum value of the communication range (CR_{min} and CR_{max}).

In order to evaluate the control system, a Matlab-based simulation tool has been implemented (see Figure 5.2) to facilitate the design and validation without considering the underlying IoT protocols. The simulation tool is able to configure the control system parameters and the network parameters such as the number of the nodes to be deployed in the network, the size of the area where sensor nodes will be deployed, etc.

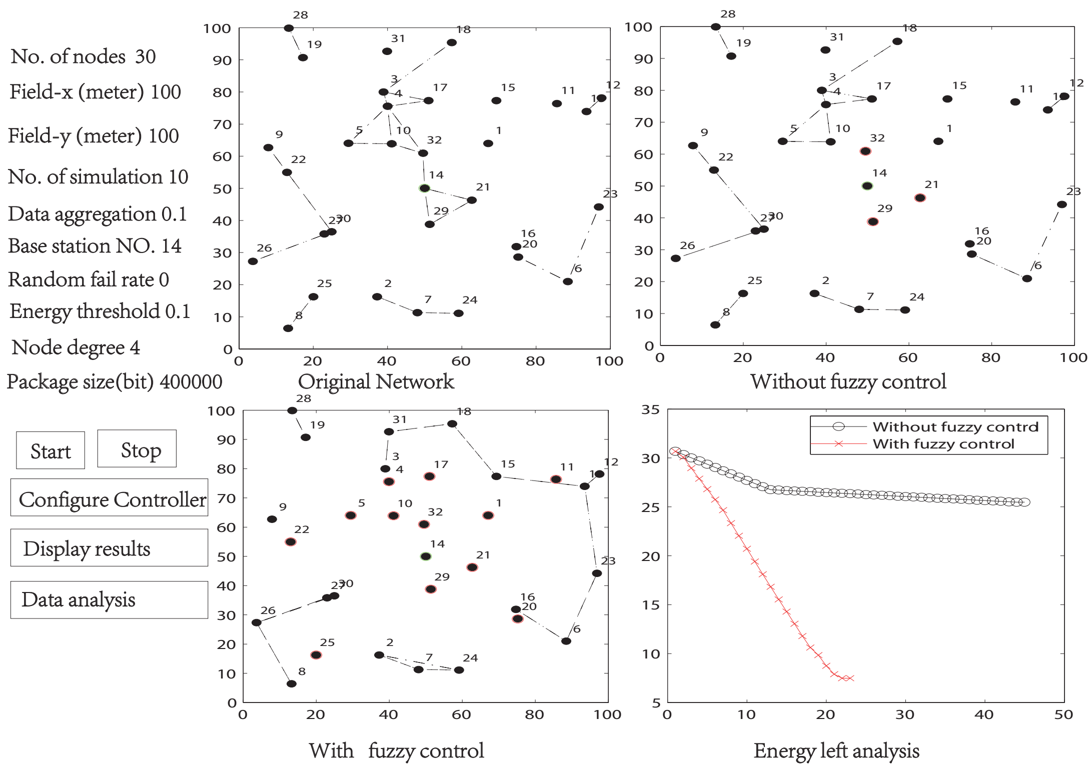

No. of nodes 30
Field-x (meter) 100
Field-y (meter) 100
No. of simulation 10
Data aggregation 0.1
Base station NO. 14
Random fail rate 0
Energy threshold 0.1
Node degree 4
Package size(bit) 400000

Figure 5.2: *Matlab-based simulation tool.*

It also visualizes the simulation process and analyzes the results once the simulations are finished.

In order to evaluate the performance of the control system, the simulation-based experimental platform were set up and configured as follows: (1) 32 sensor nodes randomly are deployed in a $100 \times 100m^2$ area. The Base Station (BS) is always located at the centre of the deployment area. (2) Each sensor node was randomly assigned an initial communication range with values within configurable boundaries, e.g. [10, 30]m. (3) The node's battery is fully charged at the beginning of each simulation. (4) The deployed nodes start transmitting packets to the BS if there is a routing path available. If there are more than one path to the BS, then the node chooses the shortest path. (5) The sensor stops sending packet when it runs out of battery. (6) The whole simulation is terminated when there are no packets received at the BS side.

With the aim of comparing the network performance, the simulation tool simulates two times on the same network topology, with the same configuration parameters: the first time without control algorithm and the second time with the control algorithms on each node, e.g. control loops based on Fuzzy Logic. The sub-figures at top-left, top-right and bottom-left in Figure 5.2 show respectively the original deployment of the nodes, the links and status of the nodes once the simulation has ended when no control has been carried out and when the control algorithm has been running. In those

Table 5.1: *Comparison of received packets at BS.*

Controller Parameters	Communication Range	P_{BS_NC}	P_{BS_WC}	Improvement
$\overline{ND} = 4$, $k_{\Delta nd} = 3$, $k_{cr} = 2$	R = [10,30],$R_{BS} = 25$	189.8	411.9	+117.02%
	R = [15,35],$R_{BS} = 30$	356.2	454.8	+27.68%
	R = [20,40],$R_{BS} = 35$	538	576.8	+7.17%
$\overline{ND} = 4$, $k_{\Delta nd} = 2$, $k_{cr} = 1$	R = [10,30],$R_{BS} = 25$	224.6	479.2	+113.36%
	R = [15,35],$R_{BS} = 30$	442.1	537.8	+21.65%
	R = [20,40],$R_{BS} = 35$	537.5	601.3	+11.87%
$\overline{ND} = 3$, $k_{\Delta nd} = 2$, $k_{cr} = 1$	R = [10,30],$R_{BS} = 25$	224.9	384.7	+71.05%
	R = [15,35],$R_{BS} = 30$	388.9	496.1	+27.56%
	R = [20,40],$R_{BS} = 35$	519.1	504.7	-2.77%

three sub-figures, the x and y axis indicate the width and the length of the field. The energy consumption due to the processing of the control algorithm has been considered negligible. The energy analysis sub-figure at bottom right in Figure 5.2 illustrates the remaining energy of all nodes at each round.

In order to measure the network performance, the total number of packets received at BS (P_{BS}) during the whole simulation has been measured. For each set of the configuration parameters, 10 different network topologies have been simulated. The nodes for each network are randomly deployed and the simulation has been executed two times: with and without the fuzzy control-based algorithm. The results are shown in Table 5.1 where the first column is the controller parameters. The second column contains the communication range interval for each node and the communication range for the BS. The third and fourth columns are the average value of P_{BS} obtained from the 10 networks deployed with the same set of parameters. The last column is the improvement percentage between the performance of the network with the control algorithm (P_{BS_WC}) and without it (P_{BS_NC}). It is calculated as:

$$100\% \times (P_{BS\_WC} - P_{BS\_NC})/P_{BS\_NC}.$$

After an analysis of the results in the table, the following observations can be made: (1) in general, more packets were received at BS when the fuzzy control algorithm was applied; (2) the number of packets a BS receives increases directly proportional with the node degree; (3) in general the self-adaptive communication range based on fuzzy control loops improves the connectivity of the network; and (4) as shown in the first sub-row of each row, both the radius of regular nodes and BS are smallest, which would lead to worsen connections at initial node deployment, but the improvement is higher. It implies that the network shows the improvement of the adaptive performance when network is not well connected at the beginning.

5.2 Control system parameter optimization

In order to improve performance, any controller design for the IoT involves the optimization of some parameters, either by experienced experts or applying some optimization algorithms. This section has leveraged three algorithms, Cross Entropy (CE), Particle Swarm Optimization (PSO), and Differential Evolution (DE), to find out the optimal configurations of parameters for the simple control system designed. Lots of works have been done in this field. To name but a few of them, [48][150] present Particle Swarm Optimization (PSO) to optimize Proportional-Integral-Derivative (PID) controller, while others use different algorithms to optimize PID, such as Genetic Algorithm (GA) [75], Simulated Annealing (SA) [67], Evolutionary Algorithm (EA) [61], etc.

5.2.1 A lightweight control system design

A simpler controller than Figure 5.1 with two loops is presented in this section to control the transmission power of each sensor node; at the same time, an appropriate objective function is defined, which is used as the target in the off-line optimization process, and afterwards the optimized controller is tested on the same network where all nodes run in an asynchronous manner.

The packets received at the BS, to some extent, reflect the network connectivity, because better network connectivity leads to more packets received at the BS. However, more packets received at the BS require more energy to transmit and process sensor data. Hence, the key is to control the network topology by adjusting the transmission power of sensor nodes at runtime. It is expected that the controller can achieve the global performance, e.g. the best tradeoff between the packet reception and power consumption. The goal can be realized by appropriately selecting the configurations of the controller parameters, e.g. initial inputs and the gains in the PID controller. The optimization problem can be generalized as a combinatorial optimization problem. There are lots of stochastic and bio-inspired algorithms can accelerate the search speed to find out the optimal configurations [161]. They are proved to be very efficient when

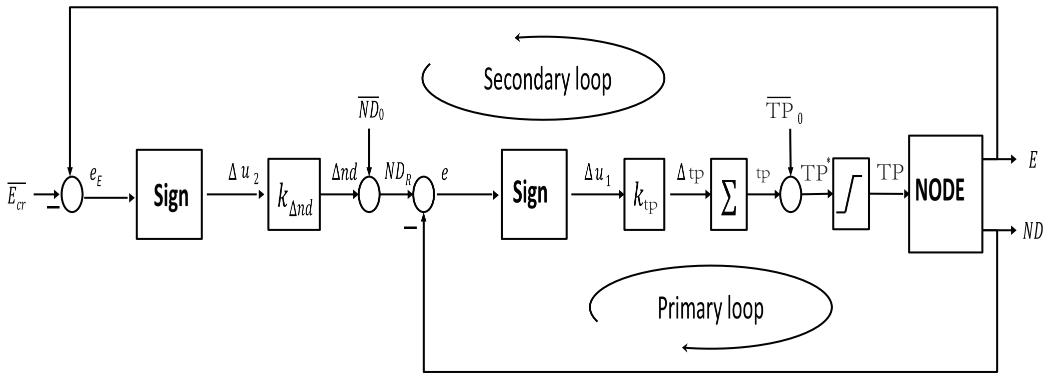


Figure 5.3: *Simplified control system design.*

the search spaces are huge, which is the case in the thesis. This thesis chooses three heuristic algorithms, Cross Entropy (CE), Particle Swarm Optimization (PSO), and Differential Evolution (DE), to optimize some parameters in the control system to achieve the tradeoff.

In practice, the inner state values of the control system are discrete and integers. For the first loop, the transmission power (TP) is discrete value in dBm unit, e.g. ranging from 0 to -31 dBm for some sensor nodes; as for the second loop, the output, Δnd, to adjust the reference node degree also has to be integer as the number of neighbors can only be positive integer or 0.

Furthermore, our work [63] intended to use the fuzzy-logic control to make the decision, but now an attempt is done to further simplify the design. As mentioned, the inner states accept only integer, so the errors e_E and e_{TP} are pre-processed before pass them to the proportional gain, $k_{\Delta nd}$ and k_{tp}. A function called "Sign" is used to simply convert the error to be -1, +1, or 0. In addition, the output of the first loop, Δtp, is the increase of the power transmission, and the result is limited by the defined constraints before applying it to the sensor nodes, as Figure 5.3 illustrated.

$$Sign = \begin{cases} +1, & input < 0 \\ -1, & input > 0 \\ 0, & input = 0 \end{cases}$$

The control system in the proposal is relatively simple, and easy to be implemented. Therefore, it requires very little computational resources to complete the two control loops, which shortens the optimization process.

5.2.2 Parameters to optimize

Many parameters in the control system shown in Figure 5.3 are unknown for a specific deployment. In this section, three methods will be presented to optimize some parameters in the control loops, according to the objective function.

In the control system, the parameters involved are $\overline{E_{cr}}$, $k_{\Delta nd}$, $\overline{ND_0}$, k_{tp}, $\overline{TP_0}$. Two of them, $k_{\Delta nd}$ and k_{tp}, are used to control the system response speed; and the others are initial conditions that depend on the deployment of nodes, such as the number of nodes in the field, their locations, distance to the BS, states of battery levels, etc. More importantly, the initial node degree $\overline{ND_0}$ of each sensor node should be different. A node which is closer to the BS should have higher $\overline{ND_0}$ because it is responsible for relaying packets from other nodes which are far away from the BS, hence the value of $\overline{ND_0}$ may end up affecting the packets received at the BS if it is too low at the beginning.

Three parameters that highly depend on initial conditions are chosen to be optimized, e.g. $\overline{E_{cr}}$, $\overline{ND_0}$, $\overline{TP_0}$, so altogether there are $n \times 3$ parameters needed to be optimized. They are put in the parameter matrix $\boldsymbol{S}$.

$$\boldsymbol{S} = \begin{pmatrix} \overline{E_{cr}}^{(1)} & \overline{ND_0}^{(1)} & \overline{TP_0}^{(1)} \\ \overline{E_{cr}}^{(2)} & \overline{ND_0}^{(2)} & \overline{TP_0}^{(2)} \\ \vdots & \vdots & \vdots \\ \overline{E_{cr}}^{(n)} & \overline{ND_0}^{(n)} & \overline{TP_0}^{(n)} \end{pmatrix}$$

where,

- n: the number of nodes in the network.
- $\overline{E_{cr}}^{(i)}$: critical energy level for node i.
- $\overline{ND_0}^{(i)}$: initial node degree reference for node i.
- $\overline{TP_0}^{(i)}$: initial transmission power for node i.

5.2.3 Objective function

Network designers expect that the nodes in IoT can collect as many sensor measurements as possible, but at the lowest energy cost. There is a tradeoff between the energy consumption and the packets received at the BS. Therefore, the objective of the function is minimizing energy consumption, while maximizing the packets obtained at the BS. The objective function J is formalized as equation (5.13). On the one hand, the energy consumption and the packets received have to be normalized before the optimization process because they are at different scales. On the other hand, the lifetime of a network is defined as the BS no longer receives packets from sensor nodes. The purpose of the optimization algorithms is finding the configurations for $\boldsymbol{S}$ that can minimize J.

$$J = w_E \cdot \alpha_E \cdot E - w_P \cdot \alpha_P \cdot P, \tag{5.13}$$

- P: total packets received at the Base Station (BS) during the lifetime of networks.

- E: total energy consumed by the network, during the lifetime of networks.
- w_E, w_P: weight factors, and $w_E + w_P = 1$.
- α_E, α_P: normalization factors.

The normalization factors are used to normalize E and P, and defined as equations (5.14) and (5.15), respectively:

$$\alpha_E = \frac{1}{\sum_{i=1}^{n} E_i^{max} N_i}, \tag{5.14}$$

$$\alpha_P = \frac{1}{P^{max} N}, \tag{5.15}$$

- N: the number of rounds when the BS no longer receives packets from any other nodes.
- N_i: the number of rounds for node i, so $0 \leq N_i \leq N$.
- $E_{i,r}$: the energy consumption for node i at the round r.
- E_i^{max}: the maximum energy consumption for node i, so $E_i^{max} = max\{E_{i,1}, E_{i,2}, \cdots, E_{i,N_i}\}$.
- P_r: The number of packets received at BS at round r.
- P^{max}: the maximum number of packets received at BS, so $P^{max} = max\{P_1, P_2, \cdots, P_N\}$.

The normalization aims at confining P and E in a reasonable value ranging between [0, 1]. The objective function J minimizes the normalized P and E, rather than P and E directly.

Every sensor node sends certain amount of packets to the BS, and updates its transmission power afterwards. This is called one simulation "round". The number of rounds for a node is the total rounds it runs until its battery becomes depleted, or the lifetime of the network ends. The number of rounds for a network, N, is the total number of rounds during the lifetime of network.

The transmission power updating can work in asynchronous or synchronous manner. By asynchronous manner, it implies that each node updates its transmission power without knowing other nodes' status. In contrast, it is synchronous when all nodes update their transmission powers at exactly the same time. In practice, asynchronous transmission power updating is more likely to happen, because it does not require operations to synchronize all nodes.

The simulations have two stages: first, control parameters are tuned off-line using the optimization algorithms; second, the optimized parameters learned in the first stage are applied to simulate the same network again and evaluate the performances. During the first stage, the asynchronous manner makes the behavior of each node unpredictable, and thus the results are not reproducible, which makes the optimization algorithms less likely to converge. Therefore, the nodes are synchronized during the first stage, but the synchronization constraint at the second stage is released, which implies that the sensor nodes are unsynchronized to simulate the network with the optimized parameters obtained in the first stage.

5.3 Optimization algorithms

The number of parameters and the network size n make search spaces big. Apart from this, the optimization process can only be applied to a specific deployment and network model, thus it can not be generalized to other networks when some conditions in the model are different. Unfortunately, there are many reasons that easily cause one network differs from other networks, for example: (1) the environment where sensor nodes deployed could be very complex so that it will impact on the real transmission range model; (2) different routing protocols perhaps choose different paths to forward data, hence affecting power consumption model; (3) the locations of sensor nodes are uncertain. Nevertheless, the optimization method itself is very generic once the nodes are deployed and the network model is set up.

The optimization problem is a complex combinatorial optimization problem. The objective function J is nonlinear and non-differentiable, so there is no direct approach to find the minimum for J. Advanced heuristic algorithms are good candidates to help finding the optimal configurations. Three heuristic algorithms, Cross Entropy(CE), Particle Swarm Optimization (PSO), and Differential Evolution (DE) are implemented in this thesis to optimize the parameters to get the minimum values for the objective function J, although they can not guarantee it is a global minimum. The implementations of CE, PSO and DE adapted to the problem of this thesis are given in this section, but for more details of the algorithms themselves one can refer to [45][121][99]. The performances will be evaluated in Section 5.4.

CE [39] is a very general stochastic optimization framework that can be utilized to solve combinatorial optimization problems. Instead of trying all possible combinations of parameters, CE leverages a randomized but efficient way to find the optimum.

Algorithm 5.3.1. <u>**CE algorithm**</u>

```
1: Initialize: Initialize $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{\sigma}}_0$. Set $t := 0$.
2: Repeat:
3: Generate Samples: Increase $t$ by 1. Generate a random sample $\boldsymbol{S_1}, \boldsymbol{S_2}, \cdots, \boldsymbol{S_D}$
   from the normal distribution $N(\boldsymbol{\mu_{t-1}}, \boldsymbol{\sigma^2_{t-1}})$, but discard and re-generate the
   samples that violate the limitations.
```

```
4: Evaluate: evaluate each set of parameter (matrix $\boldsymbol{S_i}$), and calculate the cost
   function $J$ for each set and store it in a vector $\boldsymbol{J}$.
5: Sort: sort elements for $\boldsymbol{J}$ in ascending order and corresponding indices are put into
   vector $\boldsymbol{I}$.
6: Select: Let $\boldsymbol{I_{elite}}$ be the subset of $\boldsymbol{I}$ which has the top $D^{elite}$ elements.
7: Update:
   $\hat{\mu_t} := \sum_{i \in I_{elite}} \frac{\boldsymbol{S_i}}{D^{elite}}$
   $\hat{\sigma_t}^2 := \sum_{i \in I_{elite}} \frac{(\boldsymbol{S_i} - \hat{\mu_t})^2}{D^{elite}}$
8: Smooth:
   $\hat{\mu_t} = \alpha \cdot \hat{\mu_t} + (1 - \alpha) \cdot \hat{\mu_{t-1}}$
   $\hat{\sigma_t} = \beta_t \cdot \hat{\sigma_t} + (1 - \beta_t) \cdot \hat{\sigma_{t-1}}$
   $\beta_t := \beta - \beta \cdot (1 - \frac{1}{t})^q$
9: Until: $\sigma_t < \varepsilon$ or $t > iterationMax$
   Note: $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\sigma}}$, $\boldsymbol{S_i}$ are all matrices with size $n \times 3$.
```

Algorithm 5.3.2. **PSO algorithm**

```
1: Initialize the position $present_i$ and speed $v_i$ for each particle, $i \in 1, 2, \cdots, n$.
2: repeat
3:   while $i \in 1, 2, \cdots, n$ do
4:     calculate $J_i$;
5:     if $J_i < pBest_i$ then
6:       $pBest_i = J_i$;
7:     end if
8:   end while
9:   while $i \in 1, 2, \cdots, n$ do
10:    $gBest_i = min \ \{ \ pBest_k \mid k \ is \ neighbor \ of \ i \}$
11:    $v_i = v_i + c1 \cdot rand() \cdot (pbest_i - present_i) + c2 \cdot rand() \cdot (gbest_i - present_i)$
12:    $present_i = persent_i + v_i$
13:  end while
14: until Maximum iterations or minimum error criteria is not attained
```

Algorithm 5.3.3. **DE algorithm**

```
1: Note: $D$ is the real parameters needed to be optimized; $N$ is the population, and
   $N \geq 4$; $G$ is the generation; $x_{i,G} = [x_{1,i,G}, x_{2,i,G}, \cdots, x_{D,i,G}]$; $i \in 1, 2, \cdots, N$;
2: Define upper and lower bounds for each parameter $x_j^L < x_{j,i,1} < x_j^U$
3: Randomly select the initial parameter values uniformly on the intervals $[x_j^L, x_j^U]$
4: repeat
5:   while $i \in 1, 2, \cdots, n$ do
6:     randomly select three vectors $x_{r1,G}, x_{r2,G}, x_{r3,G}$, such that $i, r1, r2, r3$ are
       distinct;
```

```
7:        $v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G})$, where $F \in [0, 2]$;
8:        $RAND_{i,j} \sim U[0, 1]$, $I_{rand}$ is random integer from $[1, 2, \cdots, D]$;
9:        **if** $RAND_{i,j} < CR$ or $j = I_{rand}$ **then**
10:           $u_{j,i,G+1} = v_{j,i,G+1}$;
11:       **end if**
12:       **if** $RAND_{i,j} \geq CR$ and $j \neq I_{rand}$ **then**
13:           $u_{j,i,G+1} = x_{j,i,G}$;
14:       **end if**
15:       **if** $f(u_{i,G+1}) \leq f(x_{i,G})$ **then**
16:           $x_{i,G+1} = u_{i,G+1}$;
17:       **else**
18:           $x_{i,G+1} = x_{i,G}$
19:       **end if**
20:   **end while**
21: **until** Mutation, recombination and selection continue until some stopping criterion
    is reached
```

In short, the parameters used in the simulation for each algorithm are:

- CE: $D^{elite} = 10$, $D = 60$, $\alpha = 0.8$, $\beta = 0.7$, $q = 5$, $\varepsilon = 10^{-3}$, $iterationMax = 30$.
- PSO: number of quality in bird: $3 \times n$; number of particle: $3 \times$ number of quality in bird; maximum number of iterations: 30; velocity clamping factor: 2; cognitive constant: 2; social constant: 2; minimum inertia weight: 2; maximum inertia weight: 0.9.
- DE: number of parameters: $3 \times n$; number of population members: $4 \times$ number of parameters; maximum number of iterations: 30; F: 0.9; crossover probability (CR): 0.9; strategy: rand/1/bin.

5.4 Simulations and validation

This section evaluates the performances for three algorithms separately with the configurations for each algorithm as provided in Section 5.3. In practice, mainly due to the simplicity of implementation and reduction of network operation cost, sensor nodes usually work in asynchronous manner. However, asynchronism brings a certain degree of randomness which is a challenge for optimization process as the optimization process is very hard to converge. To avoid the random behaviors of sensor nodes, the optimization process is performed in the condition that sensor nodes update their transmission power synchronously, as oppose to asynchronously. Once the parameters are learnt, they are applied to the network so that sensors will work in an asynchronous manner again.

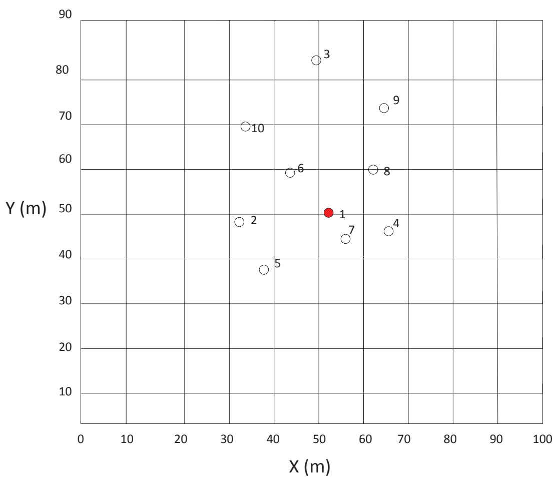


Figure 5.4: *Network topology.*

◇ Network Model

The optimization algorithms in the thesis are actually very general, so they can be applied to many network models. In case that any of those configuration changes, e.g. the number of nodes in the field, the optimization process has to be executed again to obtain the new optimized parameters. In this section, the network model adopted in this thesis will be introduced. In general, the network models used in this thesis are relatively practical.

In the simulation, there are 10 nodes deployed in the field, with node No.1 (the red node) as the BS, as shown in Figure 5.4. Each node periodically sends packets to the BS via a routing protocol. It is called one simulation "round" for a node when it has finished sending 400000 bits packets and then immediately updates its transmission power according to the decision made by the control system. Notice that the node can not update the transmission power during the time sending packets. Every node in the network, except the BS, is allowed to change its transmission power from -32 dBm to 0 dBm, according to the specification of CC2420, an IEEE 802.15.4 compliant RF transceiver, which is widely used for many sensor platforms (e.g. Sun SPOT). The routing algorithm calculates the shortest path to the BS.

The energy dissipation model is the same as [11]. Equations (5.16) and (5.17) represent the power consumed when a node transmits/receives L bit packets to/from another node at distance d. $E_{elec} = 50\ nJ/bit$ and $\epsilon_{amp} = 100\ pJ/bit/m^2$, which depend on the circuit and antenna design of sensor nodes. Each node is charged with 1J of energy at the beginning of the simulation. Nodes stop sending or receiving packets when there is no battery left. The simulation is terminated when there are no packets received at the BS. Note that there may be still several functional nodes at the time the simulation is terminated.

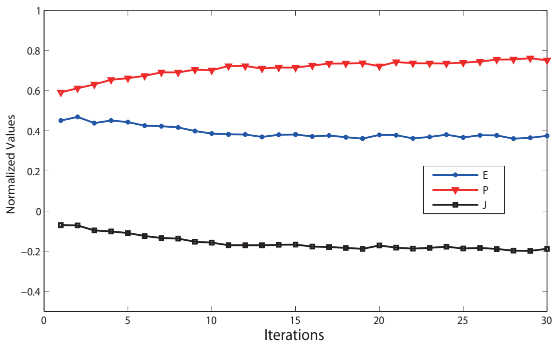


Figure 5.5: *$W_e = W_p = 0.5$, mean of J at each iteration.*

$$E_{tx} = L \times E_{elec} + L \times \epsilon_{amp} \times d^2. \tag{5.16}$$

$$E_{rcv} = L \times E_{elec}. \tag{5.17}$$

◇ **Evaluation by Computer Simulation**

At each iteration, CE generates the next set of values by using some statistics, e.g. the step 7 of the CE algorithm. First, the CE algorithm is evaluated. The mean of J, chosen from the top D^{elite} elements from the sorted $\boldsymbol{J}$ of the CE algorithm, is calculated for every iteration. As shown in Figure 5.5, when $W_e = W_p = 0.5$, CE decreases the value of objective function J as time goes by. The difference between E (total energy consumed), and P (the packet received at the BS) increases. It implies that the BS receives more packets from the whole network, but at the same time the sensor nodes consume less battery power, which is the desired performance. Furthermore, in order to show CE has the ability to adapt to different objective functions, the weights are altered. As shown in Figure 5.6, when changing the weights of E and P in the objective function, CE also can gradually reduce J, and corresponding difference between E and P increases.

To evaluate the convergence of the CE algorithm, the standard deviations of J in the step 6 of the CE algorithm are calculated for each iteration (see Figure 5.7). It is observed that the algorithm seems not to converge within the defined maximum iterations for different weights W_e. This is due to the fact that CE needs the statistics of previous iteration (the step 7) to update the new samples for the next iteration, but the network is sensitive to new samples. Another reasonable explanation is that the objective function has many local minimums, so CE is easily being stuck in local minimums.

Unlike CE, other two bio-inspired algorithms (PSO and DE) do not use the statistics to update populations for next iteration. For the sake of fairness, the best single J is

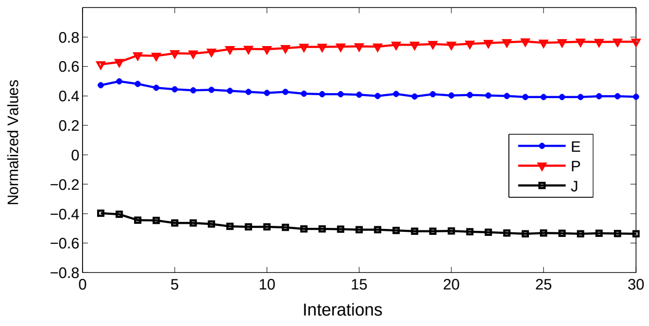


Figure 5.6: $W_e = 0.2, W_p = 0.8$, *mean values of J at each iteration.*

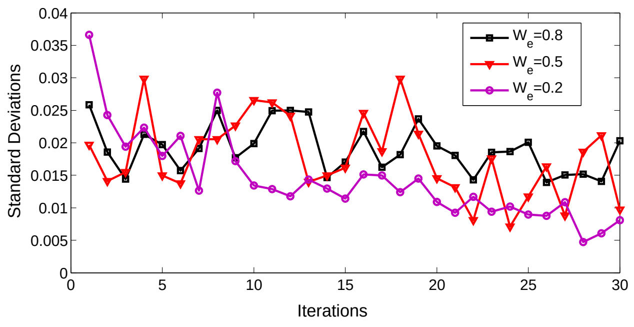


Figure 5.7: *Standard deviations for CE.*

selected in each iteration made for each of the algorithms. Figure 5.8 shows some significant results: (1) PSO and DE converge way quicker than CE; (2) CE is able to find out lower optimums than PSO and DE; (3) PSO has slightly lower optimums than DE after they converged.

The results shown in Figure 5.8 are, in practice, quite important. The computational resources can be very expensive and the optimization process can be very time-consuming, especially when the sensor network is relatively large, e.g. hundreds of nodes. In this case, PSO or DE may be applied, as they converge very fast, and much less iterations (or, time) are needed. In the case that the network is small, CE is selected prior to PSO and DE, because CE is able to make the objective function smaller.

◇ Asynchronous Network

The best control parameters obtained from Section 5.4 are applied to the same network again. However, in this case, the synchronous network constraint is removed. In the simulations, a probability is given to each node to allow it to update its transmission power at each round. By doing this, at a specific round, some nodes can update its transmission power, but the others will keep their transmission power unchanged.

Note that when the probability is 0, it means that all nodes maintain the initial transmission power during the whole lifetime of network; if the probability is 1, the

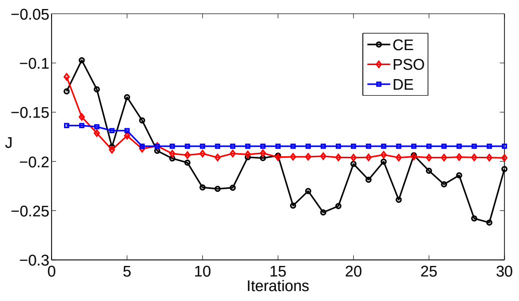


Figure 5.8: *The comparison among CE, PSO, and DE.*

results of each node should be exactly the same as in the synchronized network.

Configured with various probabilities, Figure 5.9 illustrates the J obtained with the best control parameters learnt from different optimization algorithms. The results in the figure are the average of 20 runs for each probability. As expected, when the probability is 1, the J obtained is the same as the best J in Figure 5.8. Moreover, the tendency of the three algorithms is the same: they all decrease and reach the minimum when the probability is 1. It implies that the randomness introduced by asynchronous behavior harms the network performance. The parameters that learned their behavior under the synchronous network do not work very well in the asynchronous network. This is due to the fact that the optimization process is very sensitive to network changes. In addition to that, the lower J obtained from the synchronous training process can still improve the performance in the asynchronous network in most cases.

5.5 Summary

To alleviate the implementation complexity, this thesis proposes different kinds of controller than the ones in Chapter 4. Its implementation is in Chapter 6. The controllers that proposed in this chapter are more applicable in real applications; while the ones in Chapter 4 are suitable for very large networks with nodes having reasonably rich resources.

This chapter proposes two control systems with two-loop, one with fuzzy-logic and the other without fuzzy-logic. The simulation results obtained by simulation show that the proposal improves the connectivity of networks in the presence of node failures. It is showed that the network is more resilient against node failures, which is the main goal of this work.

Moreover, based on a relatively practical network model, three algorithms are implemented and run in the simulator separately afterwards to tune the controller parameters, expected to minimize the objective function which aims at balancing the packets received at the BS and the power consumption of the IoT. More details can be

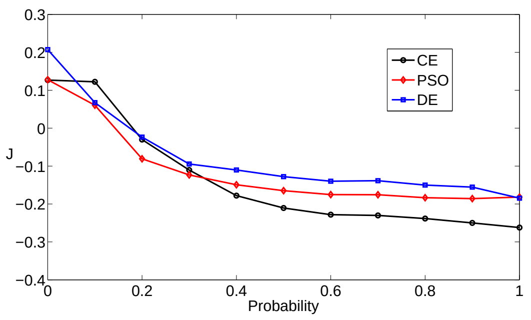


Figure 5.9: *The comparison among CE, PSO, and DE in asynchronous network.*

found in our publications [63][157].

To be more concrete, when starting with a simple controller design, the objective function that intends to make a tradeoff between packets reception and power consumption is defined. In order to minimize the objective function, there are some parameters that need to be optimized. Three algorithm, CE, PSO, and DE are adopted as the optimization strategies in the thesis. Through the simulations, it is able to find the appropriate configurations for all parameters in the controller. The simulation results show that CE can find more suitable configurations than PSO and DE within the same number of iterations; nevertheless, PSO and DE converge faster, which is good for a large network. The optimization is done off-line and in a synchronous manner. Afterwards, the optimized parameters are fed into the same network where sensor nodes run in an asynchronous manner.

The results in this chapter can be useful from a practical point of view. For a large IoT with hundreds of sensor nodes, running an optimization algorithm may take a too long time, so PSO and DE are good candidates as they converge faster. For a small network, though, the time to obtain the optimum may be not the major concern, so CE would be selected, as it can improve overall performance of the network, despite the cost of running it longer time.

It turns out that the optimization algorithm is computational-expensive, thus the best way to apply this optimization is by performing simulations first, and then applying them to a real system afterwards. Any simulation-based evaluation relies on the accuracy of the model that can map the real world devices into a simulator, which is a non-trivial task.

Chapter 6

Evaluation via Testbed

One of the proposals in Chapter 5 is going to be implemented in a real-world scenario. The implementation, in general, is the combination of close-loop feedback control and software engineering. In other words, the software application is designed and developed as a close-loop system using feedback, rather than an open-loop system which is how software is usually being developed. By doing this, adaptivity is attained and becomes an essential part of the middleware. Note that in the context of software engineering, self-adaptive is often being viewed as the non-functional requirement.

The work done is part of the "Design, Monitoring and Operation of Adaptive Networked Embedded Systems" (DEMANES) [41] project implementation, which has a target for the emerging market of applications and services for the Internet of Things (IoT). Our research group played an essential role of designing a self-adaptive architecture which employs a simplified control system (as shown in Figure 5.3).

Many partners participated in the project to solve various adaptive problems in distributed embedded systems. Nevertheless, the development of each partner is complaint with DEMANES middleware framework. This chapter only discusses our contributions to the project. Specifically, the contributions of the research group have been about applying some self-adaptive techniques to address the IoT connectivity and energy-saving problems, which have been discussed in this thesis. For more information regarding the contributions of other participants, please refer to [41].

6.1 DEMANES

The obstacle that lies in the way of designing self-adaptive networks is the lack of versatile methodologies. DEMANES, an ARTEMIS project, addresses a wide variety of challenges in the large scale adaptive embedded systems, such as designing, implementing, testing and deploying, and combines recent advances from systems and control engineering. The goal of DEMANES is to provide component-based methods, frameworks and tools for development of runtime adaptive systems, making them capable of reacting to changes in themselves, in their environment (battery state, availability and throughput of the network connection, availability of external services, etc.) and in user needs (requirements) [41]. The global objectives of the DEMANES project are:

- modeling the architecture and the operation of adaptive systems;
- supporting the design process of such systems by providing simulation and evaluation environments and test-beds;
- supporting the implementation of such system by providing services for self-organization, reconfiguration and self-optimization as parts of the execution environment;
- verifying and testing adaptive systems;
- monitoring the internal and external operational conditions and managing adaptation at runtime.

The concept, methodology and tools developed in DEMANES are to be validated and demonstrated in three different pilots: a smart urban transport, a smart airport and a smart home. The partners in the DEMANES consortium come from five countries in Europe (Italy, Spain, Finland, Netherlands, Czech Republic) and are complementary in terms of technical competencies and organizational, business and market experience. The results of DEMANES will impact the standardization on related areas (M2M, Smart Homes, 5G wireless, Vehicular Technologies, Home Automation and Security, etc.) in Europe and worldwide.

6.2 Self-adaptive software

DARPA Broad Agency Announcement (BAA) defined the self-adaptive software as [80]: "Self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible." In recent years, there is a increasing demand to incorporate self-adaptive software at runtime.

DEMANES middleware, as the name of the project hints, is one of the main contributions of this project, which defines a reference software architecture that is hardware and application independent and provides the functions, algorithms, design patterns for the adaptive embedded systems. "The DEMANES middleware supports, enhances, integrates, and hides implementation details of adaptive systems, such as reconfiguration mechanisms, information dissemination and aggregation mechanisms, and management of adaptive software components." ([41], deliverable 4.3)

6.2.1 Component-based design for DEMANES middleware

From the software application point of view, the application developers are unaware of the underlying changes. The DEMANES middleware will handle the underlying changes, e.g. failures, at runtime. The developers only have to concentrate on solving the problems of its application domain. The DEMANES middleware will provide the application developers a standard way to access the runtime environment, e.g. via library or API. The software component in the DEMANES middleware can be modified, added, removed at runtime. On the other hand, the adaptation mechanism can be modified without notifying the application. The DEMANES middleware is actually transparent to applications. To implement the DEMANES middleware, a software architecture is proposed; to facilitate the design, some tools are leveraged or developed in the DEMANES project.

After starting by identifying common functionalities in the large scale, networked, embedded system, the DEMANES middleware then provides a set of interfaces for the higher level applications, in order to create a flexible and potable system. However, when the actual implementation is taken into account, the DEMANES middleware has to be customized to the specific hardware characteristics.

The DEMANES project follows a standard methodology, including modeling, analysis, implementation, and test. The key to implement the component-based adaptive system is allowing the components to be loosely coupled; additionally, components can be modified, added, and removed at runtime, without having to shut down the system. By component, it is meant software block (task, thread, etc.), hardware resources (memory, transmission power, battery, etc.), or something as large as a whole sensor node. By doing this, the system is very flexible, and it can be easily adapted changing requirements and environments, e.g. by replacing outdated components and loading new component.

More specifically, each component in the system has to satisfy the following requirements according to the DEMANES framework:

- Firstly, components can be activated and deactivated;
- Secondly, components are parameterized;

- Thirdly, components can be rewired.

The first requirement controls the status of the components and allows them to be shown or disappear in the system; the second, can make the component controllable or configurable (e.g. sensor measuring frequency, measurements range, threshold, etc.); while the last requirement helps a component to be integrated with other components to form an arbitrary larger component which possesses more powerful capability. However, the three requirements above perhaps can not be satisfied at the same time for a specific component. For instance, the third one that needs the components to be wired together may only be feasible for some components under certain conditions, e.g. the hardware are the same and well connected.

In general, self-adaptive systems share a common characteristic, namely redundancy. There must a (some) back up component(s)/configuration(s) that can fulfill the same or similar functionality when one fails. There are many strategies that be used to implement the component-based architecture from the software point of view.

- components/tasks scheduling, e.g. start, stop, suspend. By doing this, components are controllable and tasks can be distributed among many pieces of components.
- replicating a software component in many nodes in the system and keeping only one as the active component; the others are regarded as backup components in sleep mode for the purpose of energy saving, and will afterwards be activated and take over the task. In this way, the system is able to move the task from one node to another one, thus exhibiting self-adaptive behavior.
- updating/replacing code at runtime.
- compositions: assemble small components into a larger, more powerful and meaningful component.
- reasoning: a reasoning engine makes decisions on new configurations/actions for components, e.g. transmission power, transmission rate, routing, etc.

6.2.2 Self-adaptive software techniques

Self-adaptive software is inherently interdisciplinary and highly dependent on the problem that tries to solve [117], which involve many disciplines, such as: software engineering, artificial intelligence, decision theory, control theory, network and distributed computing, and optimization theory. [81] uses metaphors to describe the self-adaptive software: coding an application as a dynamic planning system; coding an application as a control system; coding an application as a self-awareness system.

One way to implement self-adaptive software is modeling the software in an adaptation process. The DEMANES middleware takes into consideration the adaptation process as the core of its design by modularizing all components. To name some software frameworks/architectures, Model-Integration Computing (MIC), Architectural Description Languages (ADL), Attribute-Based Architecture Styles (ABAS), Component-Based Software Engineering (CBSE) are popular. To support self-adaptive software realization two notable techniques, Aspect-Oriented Programming (AOP) and Service-Oriented Architecture (SOA), are often good ones. In particular, SOA has become a very popular web-service-based architecture that facilitates the design of self-adaptive software due to its flexibility for composition, orchestration, and choreography. In the DEMANES project and DEMANES middleware, the SOA architecture is used as a common technique to integrate platforms among partners, and also as a service composition framework for dynamic adaptation. SOA will be further discussed in Section 6.2, because it is part of the DEMANES middleware.

Table 6.1: *Self-adaptive software techniques.*

Discipline	**Techniques**	**Remark**
software engineering	MIC, ADL, ABAS, CBSE, AOP, SOA	They are software modeling methods
artificial intelligence	machine learning, agent-based, AI planning, decision theory, utility theory	Make the system intelligent in response to changes
control theory	PID, fuzzy logic, etc.	Tune the parameters in response to changes
network and distributed computing	policy-based, rules-based, QoS, middleware, virtualization	Provide guidance for decisions and actions, and environment for others techniques
optimization theory	bio-inspired algorithm: evolutionary algorithm, genetic algorithm, etc.	Optimize parameters and performances

Table 6.1 summarizes the main techniques used in each discipline. The techniques that are leveraged to develop the self-adaptive system are: (1) CBSE/SOA in the software engineering category; (2) PID and fuzzy-logic in the control theory category; (3) parameters optimization in the optimization theory; and, (4) apparently a middleware-based approach in the network/distributed computing category. In the

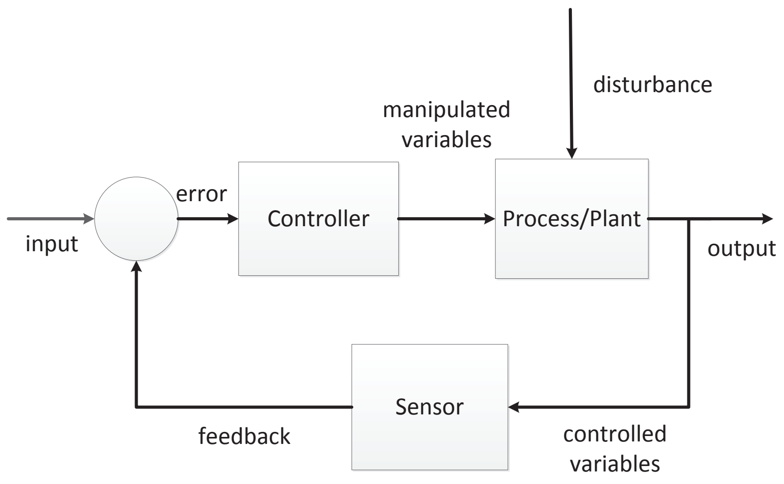


Figure 6.1: *Feedback control for the self-adaptive software.*

next pages, the details of the techniques that have been used in the project will be introduced.

◇ **From the control theory point of view**

To some extent, lots of systems (software and hardware), which possess self-adaptive nature, can be modeled as a feedback control system (e.g. Figure 6.1), like automation computing or self-management. In fact, the terms self-adaptive, automatic computing, and self-management are sometimes used interchangeably. It is difficult to draw a clear distinction among those. But in general, automatic computing has broader senses and applications, while self-adaptive software falls into the umbrella of automatic computing, because self-adaptive is limited at the application or the middleware layer, while automatic computing can be placed down to the underlying layer, such as network and physic layer. The self-adaptive software can be further broke down to several "self-*" properties [117]: self-reconfiguration, self-repairing, self-optimizing, self-healing, self-awareness, etc.

◇ **From the software engineering point of view**

Instead of modeling the control system as a feedback control system like Figure 6.1, a similar perspective can be drawn as depicted in Figure 6.2, which is called the MAPE-K loop, in the context of autonomous computing. It includes Monitoring, Analyzing, Planning and Executing functions, with the addition of a shared Knowledge. The Monitoring process is responsible for monitoring the status of the system and detecting the changes/abnormalities. The Analyzing process decides when to response to the

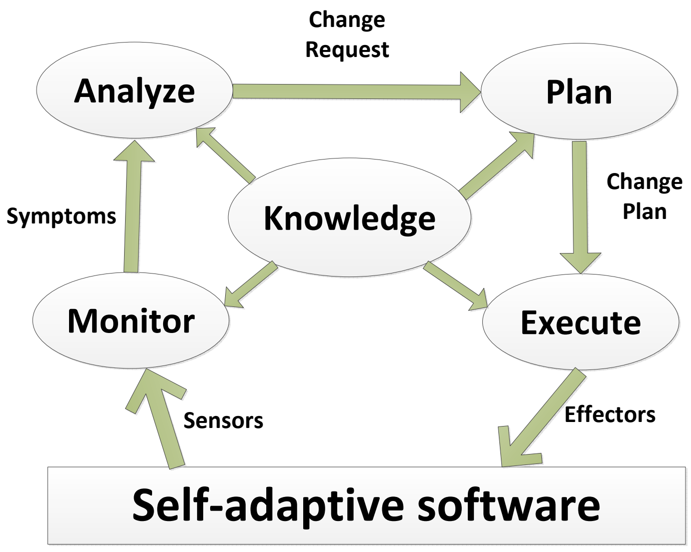


Figure 6.2: *Self-adaptive architecture.*

changes. The Planning process decides what needs to be changed. The Executing process takes the actions determined by the Planning process.

Later, in the implementation section (Section 6.2), Figure 6.2 will be further adapted to the software components that are implementable, see Figure 6.3.

In the context of self-adaptive software, the monitoring process is done by the software "sensors". The software sensors can be in various forms [117], e.g. log, event notification models, management protocols, design patterns, call-back functions, etc. It is worth noting that some of the sensors are based on the design patterns that allow changing software component at runtime, e.g. wrapper, proxy, strategy pattern. Some of the sensors are implemented in the middleware layer, like what the DEMANES middleware does.

On the other hand, the actions can be taken in the context of self-adaptive software have various forms as well. The relatively simple ones are: caching data to lower the response time; changing data quality, or compress data, or change data type to save bandwidth; adjusting parameters to meet goals; changing the aspect of a component to another one with different quality; changing algorithm to another, etc. Those methods are generally simple as they only just involve tuning some parameters. There are some more complex actions, though: replacing a component with another of different quality; changing/reconstructing the architectures of the system; provisioning additional resources; restarting or redeployment, etc.

Self-adaptivity can be introduced into software systems using two strategies: at the developing stage, or at the runtime. The latter needs the system to have the learning capability, or be capable of evolving, or intelligence (e.g. by machine learning).

It is expected that self-adaptive behaviors can be managed at runtime without human involvement, rather than hard-coded which needs to have the software recompiled again. Moreover, it is desired that new behaviors can be introduced at runtime, because it is very likely that some functionalities are overlooked at the design stage.

Particularly, the reasoner is the core part, which makes the decision on when and how to reconfigure the system. It can be located in various implementations, e.g. state machine, rules, conventional PID control, advanced control like fuzzy logic control, or machine learning, etc.

The implementation of the self-adaptive networks follows the DEMANES middleware framework. Consequently, the reasoner is the implementation of fuzzy-logic control (Figure 4.2, 4.3, 5.1), or a simplified controller as in Figure 5.3. Due to its simplicity and effectiveness, only the structure of the control system in Figure 5.3 is implemented. Others are tested in the simulator. Furthermore, because of the iterative nature of the optimization algorithms, the optimization process is implemented and validated in the simulator as well.

6.3 Middleware implementation

In this section, the controller implementation in the DEMANES middleware is discussed. In general, there are two parts: the control system implementation from the software perspective and the SOA implementation which serves as an integration platform for all participants in this project; SOA itself is a self-adaptive framework that supports runtime reconfiguration.

6.3.1 Control system implementation

The control system in Figure 5.3 is mapped into implementable software components described in Figure 6.3 and the corresponding class diagram is described in Figure 6.4, which follows the DEMANES frameworks. Other reasoning mechanisms, such as rule-based reasoning and machine learning, can also be adapted to this framework.

More specifically, the reasoning engine consists of the following parts:

- System under adaptation: this is the target system that needs self-adaptive control, e.g. sensor nodes;
- Observer: observe the changes in the system; usually it comprises the variables/resources/components that need to be controlled;
- Actuator: the part that can make action according to the output(s) of the reasoning engine;
- Reasoner: the key component that implements the decision-making algorithm;

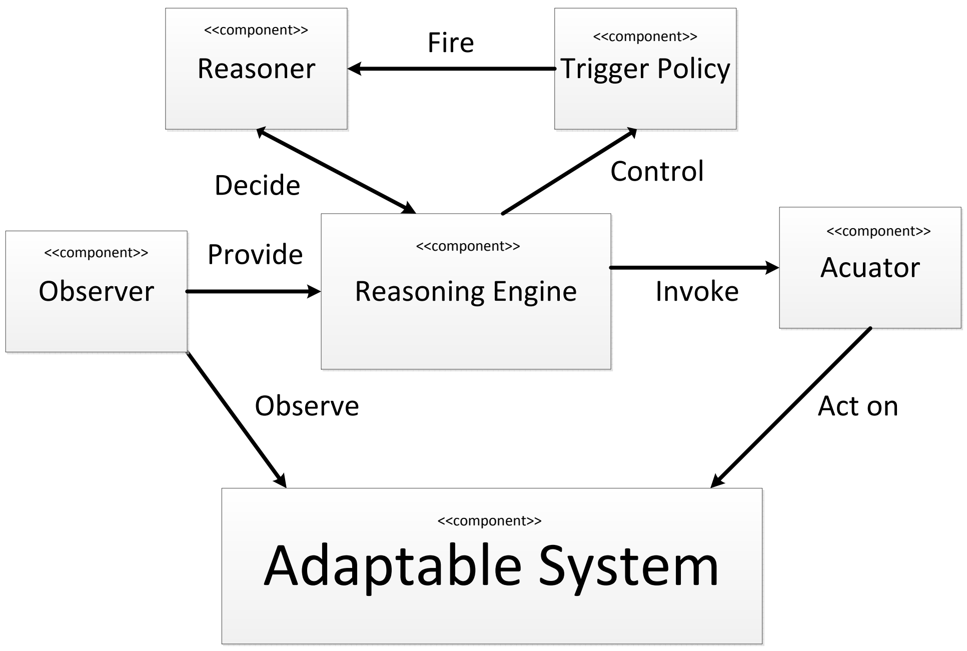


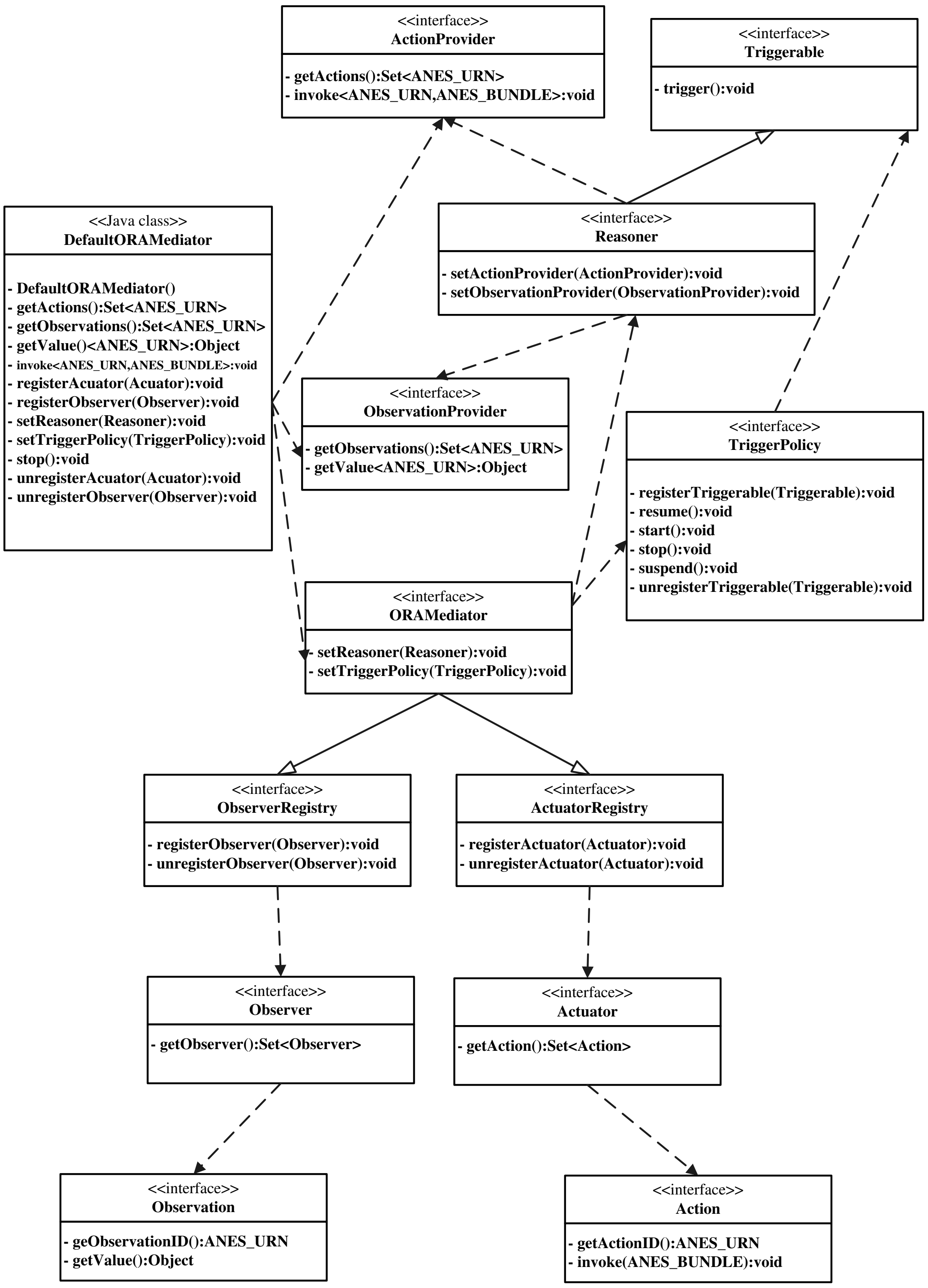
Figure 6.3: *The control system implementation in DEMANES middleware.*

- Trigger policy: decide when or under which condition(s) the reasoner starts the reasoning process;

- Reasoning engine: the mediator that organizes all components above.

Unlike a typical control system, this design possesses an unique characteristic: all components are isolated from each other, and controlled by a central component called "Reasoning engine", which makes the reasoning system more flexible. The direct communication between two components is not allowed. All other components must be registered in the Reasoning engine before it is used, which enhances the reusability and interchangeability of the system. By doing this, the publish/subscribe software design pattern is leveraged to implement the control system. More importantly, it is allowed to have more than one components which belongs to same type to register in the Reasoning engine, e.g. more than one reasoner. In the context of software, more than one instantiations are possible.

In addition, some components can have many status, e.g. create, active, idle, stop, which make them controllable and allow operations can only be conducted when a component is under a specific status. For example, the reasoner can only access the observer when it is in active status, or the trigger policy can only fire the reasoner when the reasoner is in idle status, etc.



<<interface>> ActionProvider
- getActions():Set<ANES_URN>
- invoke<ANES_URN,ANES_BUNDLE>:void
<<interface>> Triggerable
- trigger():void
<<Java class>> DefaultORAMediator
- DefaultORAMediator()
- getActions():Set<ANES_URN>
- getObservations():Set<ANES_URN>
- getValue()<ANES_URN>:Object
- invoke<ANES_URN,ANES_BUNDLE>:void
- registerAcuator(Acuator):void
- registerObserver(Observer):void
- setReasoner(Reasoner):void
- setTriggerPolicy(TriggerPolicy):void
- stop():void
- unregisterAcuator(Acuator):void
- unregisterObserver(Observer):void
<<interface>> Reasoner
- setActionProvider(ActionProvider):void
- setObservationProvider(ObservationProvider):void
<<interface>> ObservationProvider
- getObservations():Set<ANES_URN>
- getValue<ANES_URN>:Object
<<interface>> TriggerPolicy
- registerTriggerable(Triggerable):void
- resume():void
- start():void
- stop():void
- suspend():void
- unregisterTriggerable(Triggerable):void
<<interface>> ORAMediator
- setReasoner(Reasoner):void
- setTriggerPolicy(TriggerPolicy):void
<<interface>> ObserverRegistry
- registerObserver(Observer):void
- unregisterObserver(Observer):void
<<interface>> ActuatorRegistry
- registerActuator(Actuator):void
- unregisterActuator(Actuator):void
<<interface>> Observer
- getObserver():Set<Observer>
<<interface>> Actuator
- getAction():Set<Action>
<<interface>> Observation
- geObservationID():ANES_URN
- getValue():Object
<<interface>> Action
- getActionID():ANES_URN
- invoke(ANES_BUNDLE):void

Figure 6.4: *The reconfiguration class diagram of DEMANES middleware.*

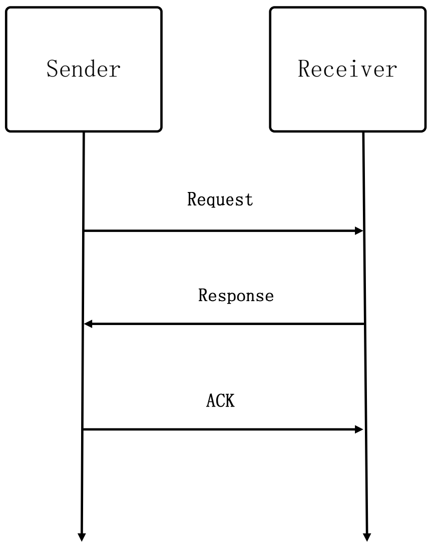


Figure 6.5: *Three-handshake neighbor discovery protocol.*

Consequently, the mechanisms in turn make the system more self-adaptive to the changing environments from the software point of view. For instance, it can register two different reasoners (or, observers, actions, etc.), and allow them to automatically change from one to another when one is stopped, without interrupting the system, which is the essence of the self-adaptive design, and certainly of the DEMANES project. The interfaces and their interactions have been clearly defined in the project.

Lastly, in order to make the parameters in the system configurable at runtime. The parameters are made public if and only if the parameters need the reconfigurability.

To make the implementation in Figure 6.3 more specific, the details are introduced here, taking into consideration the two-loop control system in Figure 5.3 and the real sensor platform called Sun SPOT [108]:

- Observer: observe the Node Degree (ND) and energy level. To obtain the ND, a node degree discovery protocol running upon the routing layer is implemented.

  The node discovery works like the three handshake fashion in TCP protocol, as shown in Figure 6.5. A node broadcasts the neighbor discovering request, any node that receives the request will response to the node which sends to it, and then an ACK message has to send from the source to acknowledge the response message. By doing this, it will make sure that if node B is in the neighbor list of node A, then node A is in the neighbor list of node B.

  To save energy and reduce the wireless inference among nodes, the node degree discovery works in an on-demand fashion. At the same time, the energy observer is trivial as it is provided by the Sun SPOT API directly.

- Actuator: the action that changes the power transmission, from -32dBm up to 0dBm which compliant with the specification of IEEE 802.15.4.
- Reasoner: execute the two control loops logic in Figure 5.3. It is worth mentioning that the two loops will possibly not run at the same time; instead, both of them will be executed under certain conditions which are defined by the trigger policy. For instance, only the primary loop will be run in some cases.
- Trigger Policy: decide when to start the reasoner, according to the errors between the observed inputs and the real ones. Some modifications have been made. Several rules have been added to fire the reasoner. Not only the errors can start the reasoner, but also some rules have been defined to fire the reasoner.
- Reasoning Engine: implement the logic that organizes all the components above. It is nothing more than a container where all the components are registered. In the Sun SPOT environment, it is the main body of the program that will be running when the application starts.

Due to the Sun SPOT is a relatively powerful sensor device, it is capable of supporting multiple threads. Consequently, this work takes advantage of that to develop each component as a thread.

6.3.2 Service-oriented architecture

"*Service-oriented architecture* (SOA) is a software design and software architecture design pattern based on distinct pieces of software providing application functionality as services to other applications. This is known as service-orientation. It is independent of any vendor, product or technology" [138]. SOA is part of the self-adaptive techniques employed in the project, and it elaborates different aspects of the self-adaptive feature of the DEMANES middleware. To complete the implementation of the DEMANES middleware, the SOA part is briefly introduced in this section.

As mentioned before, SOA is leveraged as an integration framework in the DEMANES middleware to facilitate communications among different components developed by several partners. Moreover, SOA itself is served as a good runtime reconfiguration environment, because it supports many functionalities that are very suitable for self-adaptive software (in particular, runtime reconfiguration), such as: (1) runtime updating, removing, adding components; (2) integration of different applications and (3) service-oriented architecture allows the service composition.

Among various implementations/examples of SOA, OSGi (Open Service Gateway Initiative) [109] are used both in academia and industry, which exhibit many SOA features, such as service-oriented, loose coupling, integration, and reusability.

OSGi provides a nice programming model (more precisely, dynamic module system for Java) that decouples the service providers and consumers. The structure of OSGi

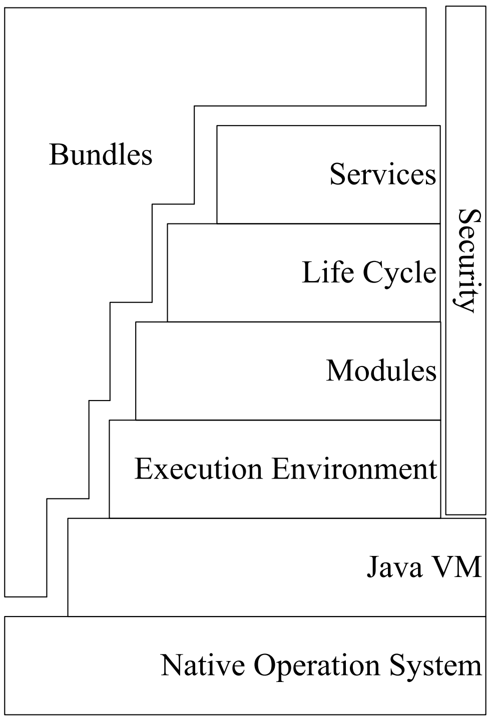


Figure 6.6: *OSGi architecture [109].*

is illustrated in Figure 6.6. OSGi implementation sits on the top of JVM and provides mechanisms for service management, execution, security and life cycle control of modules. The notion "service" in the context of OSGi is actually a Java interface. The providers implement the services, and the consumers use the services supplied by providers. However, the key is that both of them, providers and consumers, do not have to know each other. They just run their applications in a common platform, called OSGi runtime (e.g. Apache Felix, Equinox, Knoplerfish). The components deployed in OSGi runtime are called "bundle". Briefly, the bundle is nothing more than a jar file which contains annotated POJO (Plain Old Java Object) with manifest files that describe a bundle, e.g. make packets explicitly exported/imported.

6.4 Experimental results

This thesis tests the proposal with the focus on the sensor network connectivity in the presence of node failures. It is expected that the network can reach the steady status, and more importantly the network is able to maintain network connectivity when some nodes are moved away or turned off.

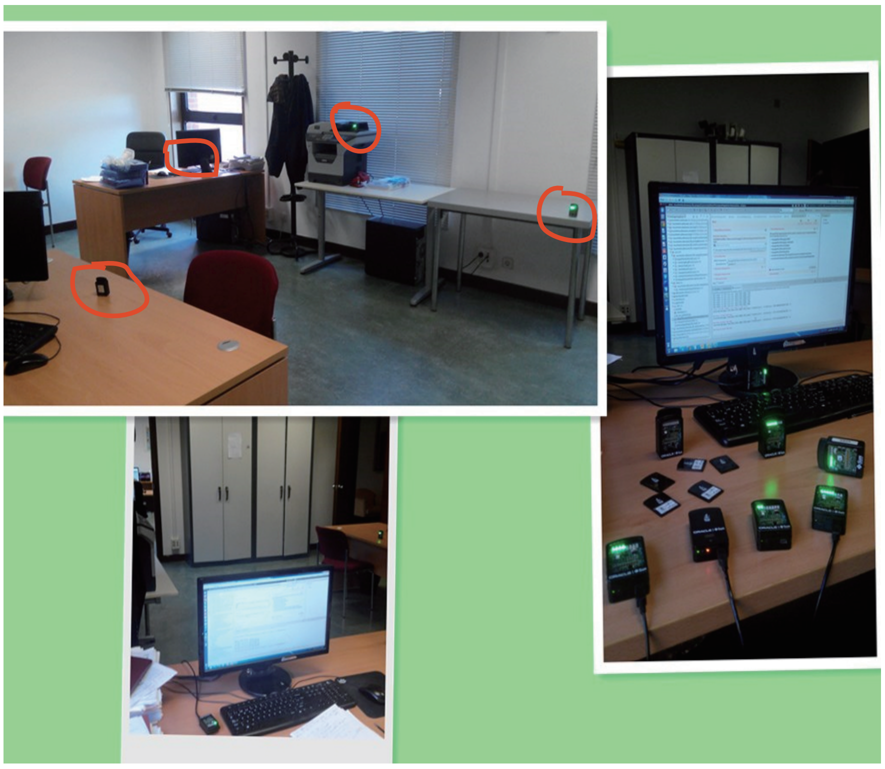


Figure 6.7: *Indoor deployment environment: the south campus at UPM.*

6.4.1 Experiment setup

The experiments are carried out at the south campus of Universidad Politécnica de Madrid (UPM), where our research center situated. Sun SPOT, the sensor deployed, is equipped with rechargeable battery, chip with IEEE 802.15.4 radio standard, analog inputs, LEDs, switches, and can measure acceleration and temperature. The routing protocol that in Sun SPOT sensors is Link Quality Routing Protocol (LQRP), which considers battery level and link quality to route the packets.

The tests are performed in both indoor and outdoor environments, respectively. The nodes are randomly deployed in the field, and automatically adjust their transmission powers based upon the controller designed. The Base station (BS) is attached to the laptop and collects data from other sensor nodes if there is a path to the BS. Hence, from the BS side, it can be known how many nodes are connected to it, which implies the performance of network connectivity.

It is worth to mention that all nodes in the network are supposed to be treated with equal importance from the graph perspective, but in reality there is a BS serving as a gateway that can bridge the sensor networks with other types of networks, e.g. Ethernet. To make the deployment more meaningful with regards to real applications, this work measures the network connectivity only from the BS side.

Every node tries to connect and sends data to the BS periodically if there is a direct

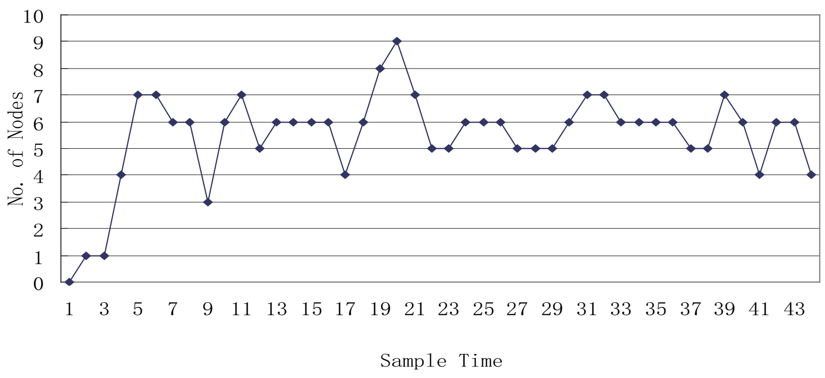


Figure 6.8: *Indoor test: the number of nodes connected to the BS.*

or indirect path exists. The packet is forwarded to the BS by LQRP. Sensor nodes are programmed to be able to relay data from other nodes if necessary. The BS is responsible for storing all data and calculating the number of nodes that is connected to it.

The configurations for the control system are: $\overline{E_{cr}}$ is 50% of maximum battery capacity (800 amps). $\overline{TP_0}$ is the minimum transmission power level, that is -32dBm. k_{tp} is set to 1 for the indoor test, and 2 for the outdoor test. $k_{\Delta nd} = 1$. $\overline{ND_0} = 3$.

6.4.2 Indoor test

The indoor test is done in the lab, as Figure 6.7 displays. There are 9 nodes placed inside the rooms. Because the rooms are rather small, a node can easily reach other nodes (maybe all other nodes) in one hop with even very low transmission power, hence there is little space for sensor nodes to apply the control theory. Therefore, for the indoor tests, this work puts emphasis on the stability of the network.

The results are depicted in Figure 6.8. Notice that the timescale at x-axis represents a 12 seconds interval. In general, the controller works as expected because of the number of nodes that can connect to the BS with relative stability. It proves that the controller, to some extent, can reach the steady point, thus it results in reliable communication among nodes, which is important for applications.

Of 9 nodes in the rooms, there are, on average, 6 nodes that can be seen at the BS, but a few nodes still lose their connection to the BS. It happens to the outdoor tests in next Section 6.4.3 as well. The reasons are twofold: one associates with the parameters in the design; another is due to the network deployment. More specifically:

(1) Nodes update their transmission power periodically (e.g. every 10 seconds in this test), so the routing path can be altered constantly. It must be taken into account that the time to establish new and reliable routing path perhaps is likely to be longer than the update of network topology. Consequently, a node is unable to find a path to the BS in time because the topology is changed too fast. For example, the route path

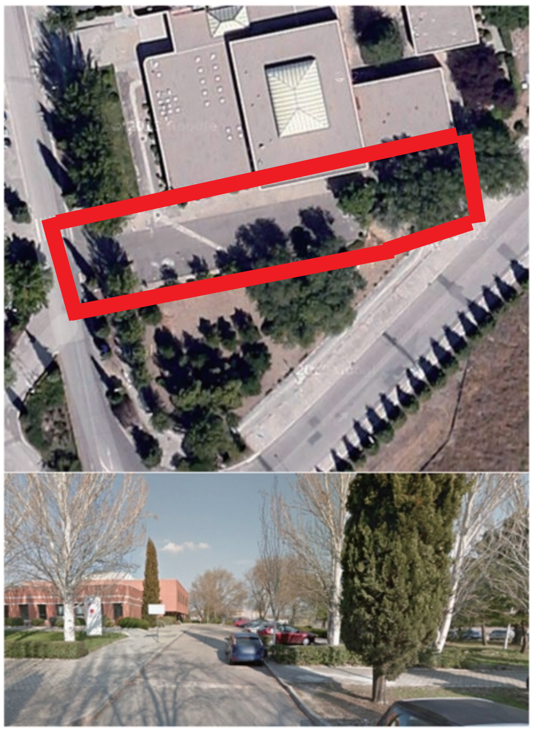


Figure 6.9: *Outdoor deployment environment: the south campus at UPM.*

disappears due to its neighbors shortening their transmission powers. This problem can be alleviated by increasing the period to update the transmission power, or execute the controller. However, longer update time brings the disadvantage that delays the routing path recovery for disconnected paths.

(2) More importantly, the results actually support the theory of this thesis. The theory that developed aims for ***large stochastic*** networks, so the theory is expected to be using, ideally, a large amount of nodes. However, with 9 nodes in the lab, it is impossible to perform large experiments. The probability that all nodes are connected (therefore can reach the BS) is low, because the nodes presented in a network are not enough. It is more likely that there are few nodes would loss the connection to others, certainly including the BS node. Theoretically, if the transmission power is big enough, or the number of nodes in the field is big enough, the likelihood that all nodes become connected together will be much higher (as proved in Chapter 3).

6.4.3 Outdoor test

The outdoor deployment environment is shown in Figure 6.9. It is a $70m \times 15m$ parking lot surrounded by a building and many trees. There were few cars at the time the experiments were conducted. Besides, there was a container, which is shown in Figure 6.10 as well. It is an ideal outdoor test environment that provides complex, but realistic, scenario where obstacles, metal materials and high buildings exist.

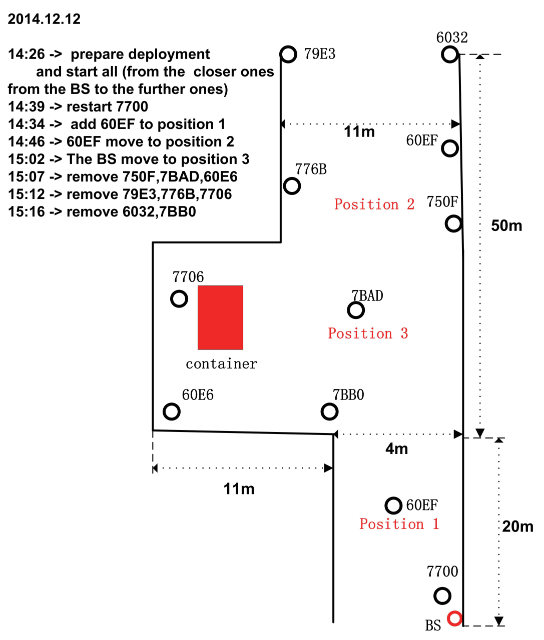

2014.12.12
14:26 -> prepare deployment and start all (from the closer ones from the BS to the further ones)
14:39 -> restart 7700
14:34 -> add 60EF to position 1
14:46 -> 60EF move to position 2
15:02 -> The BS move to position 3
15:07 -> remove 750F,7BAD,60E6
15:12 -> remove 79E3,776B,7706
15:16 -> remove 6032,7BB0
79E3
6032
11m
60EF
776B
Position 2
750F
50m
7706
7BAD
Position 3
container
60E6
7BB0
4m
11m
60EF
Position 1
20m
7700
BS

Figure 6.10: The outdoor locations of the Sun SPOT sensors.

To make the deployment more clear, the exact locations of each node are illustrated in Figure 6.10. At the top-left, the operations of the test flow are explained. The node is identified by 4 digits, which are actually the last 4 digits of the MAC address of node.

The BS is put at the bottom in the beginning. Every node tries to connect and sends data to the BS periodically if a direct or indirect path exists. Sensor nodes are programmed to be able to relay data from other nodes if necessary. The BS is responsible for storing all data and calculating the number of nodes that are connected to it.

To avoid all nodes to reach the BS in one hop as the indoor test, the reference ND for each node is set to a not too big value. This experiment sets the ND to be 3. Certainly, it may cause some nodes that are located the most far away not to reach the BS. However, it gives us the chance to see how the network behaves when some nodes fail.

The results are displayed in Figure 6.11. Again, the timescale at x-axis represents a 12 seconds interval. It is concluded that, on average, there are almost half of the nodes, namely 5 of them, that can be seen at the BS side with a total figure of 11 nodes being deployed. The most interesting part is highlighted in red color in Figure 6.11, and detailed in Figure 6.12. Three nodes are removed at 15:07 and another three nodes at 15:12. The network shows the ability to resist the loss of nodes. For instance, the total number of nodes after the first time removal operation is 8, but it shows that the number of node connected to the BS does not decrease. It is due to the fact that some sensor nodes have decided to increase the transmission power so that its ND can recover to the expected value. Similarly, after the second removal operation, the total number

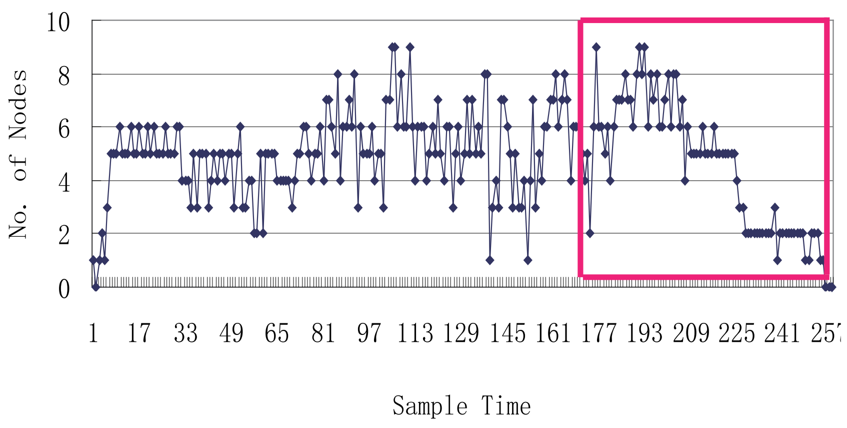


Figure 6.11: *Outdoor test: the number of nodes connected to the BS.*

of nodes in the network becomes 5, but the nodes that can connect to the BS is still about half of the remaining nodes, that is 2 nodes.

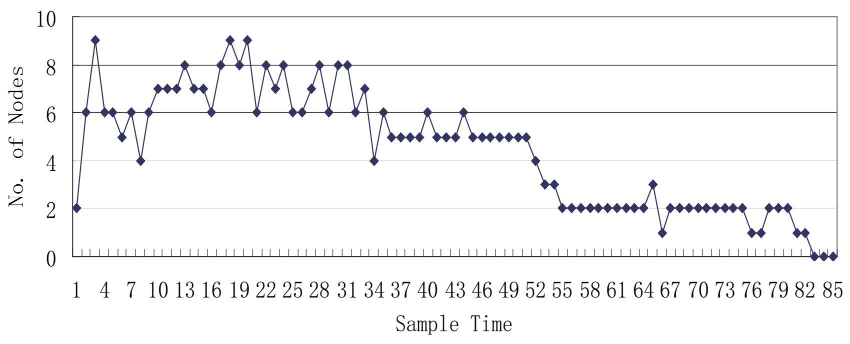


Figure 6.12: *Outdoor test: the number of nodes connected to the BS when failures occur.*

In other words, with the configuration (defined locations of nodes, and specific parameters in the controller, e.g. ND = 3 in this test), the network is able to maintain the number of nodes connected to the BS to be always roughly half of the total amount of active nodes, even when failures happen. It implies that the performance of real tests matches the theories and simulations that have been developed in the previous chapters, for example:

- As explained in Section 6.4.2, the thesis targets for large stochastic networks, hence the network connectivity is characterized by probability, which means that there is no 100% guaranteed connectivity. Nevertheless, for a specific deployment and configurations, the probability would be the same. As shown in Figure 6.12, it is always about 50% nodes being able to connected together.
- From the network resilience point of view, network connectivity and fault-tolerant

capability can be improved by tuning some configurations of the controller. It is expected that the network connectivity can be improved by altering the parameters of the controller. For instance, if the desired ND is increased, say ND = 5, then the probability that more nodes are connected together must be higher than 50% of all active nodes.

- From the applications perspective, the self-adaptive behavior that is capable of resisting node failures makes the services provided to users more reliable.

- By means of simulation, Figure 4.6 in previous Chapter 4 depicted the evaluation done on the stability of the controller; in this chapter, by means of implementation and evaluation in real scenarios, the stability of the controller is presented, and the results are shown as compliant with the simulation results.

6.5 Summary

One of the proposals in the previous chapter is evaluated in real wireless sensor networks. The implementation details and results are presented in this chapter. The self-adaptive control loop is mapped into software components, implemented in the OSGi software development environment and finally tested in indoor and outdoor scenarios.

The experimental results match the theories and simulations that were developed in the previous chapters. The experimental results demonstrate the statistical property. For instance when few nodes fail, the average number of nodes and the proportion of nodes that can be connected to the BS are relatively stable. It implies that the network is able to response to and tolerates node failure, which is the main objective of this thesis. At the time of writing this thesis, the results of this chapter were submitted to [158] as a manuscript proposal which was under review.

Chapter 7

Conclusions and Future Works

To conclude this thesis, the last chapter briefly summarizes the work that has been done and points out future works that are worth to further explore. The network resilience problems of the IoT are going to be everlasting and evolving topics, which will deserve much more effort in the future. The specific problem this thesis tried to address, namely the network connectivity, is only a small fraction among the issues many researchers have been working on. The approaches involved in the thesis could be considered as novel trials of the control and optimization techniques for the coming era of the IoT, and the results would be hopefully useful for researchers who are interested in similar problems.

7.1 Conclusions

The IoT is regarded as a large stochastic and distributed network deployed in environments where human intervention is expected to be minimum when nodes/objects/things are deployed. IoT designers and users, therefore, want that the IoT nodes are able to demonstrate self-adaptive behaviors, in the hope that nodes can automatically reconfigure themselves to achieve some goals, such as fault-tolerance and energy-efficiency. The capability of a network to adapt itself against unexpected changes (such as malicious attacks and battery depletion) brings many benefits; for instance users can obtain a guaranteed quality of service, while network maintenance costs are reduced.

In order to identify appropriate techniques to be used in this research concerning network connectivity under the circumstances when node failures take place, the thesis starts by surveying the methodologies used in the topology control domain. Topology control techniques can be applied to three different deployment stages: pre-deployment, re-deployment, and post-deployment. The techniques used in this work belong to the post-deployment stage, where the self-adaptive controller is used after nodes are deployed.

The key to keep a network resilient against failures is maintaining it $k-connected$, where k is an important indicator of IoT resilience. This thesis proves, theoretically, that adjusting the transmission power of each node to manage the node degree can be utilized to achieve a $k-connected$ network with high probability. It also proves the analytic results concerning the relationship between the probability of network connectivity and the communication range. For instance, there exists a constant number of neighbors, $C(\varepsilon) = log(\frac{1-log\varepsilon}{\varepsilon})$, for which the network becomes connected with probability increasing from $P(t_1 = \log(\frac{n}{b+1}))$ to $P(t_2 = \log(n) + b)$ (see Theorem 3.2.3). It is also proved that, by means of obtaining more neighbors, the connectivity of networks is improved, which is in line with instinct. More importantly, the resulting network becomes *k-connected* as soon as the network degree is k. One of the advantages of this theory is that it makes the implementation of the communication range control distributed without the needs of other nodes' information, which is good for large IoT as it reduces the cost of operations.

Based upon the theoretical results, the methods and models to control the transmission power of the IoT nodes are studied. By means of simulation experiments, the performance of the proposals has been evaluated, and part of them have finally been implemented in a real application supported by the European project DEMANES.

Most of self-adaptive behaviors end up with leveraging the control theory, therefore this thesis develops the fuzzy-logic and PID controller, and evaluates the performance mainly by computer simulation. Two localized and energy-efficient approaches, called LFTC and RFTC, based on the fuzzy-logic, are proposed. LFTC relies on a mathematical model, where the parameters of the fuzzy-logic controller are automatically leaned. RFTC is heuristic, so its parameters have to be tuned according to a specific deployment to achieve the best performance. Compared with other algorithms such as NONE, LTRT and List-based by simulations, LFTC and RFTC improved the capability of a network to resist against node failures by adaptively adjusting the communication range.

Moreover, considering devices are often resource-constrained in the IoT, a lightweight controller is proposed, which consists of two loops. It is shown that the network is also robust against node failures like the fuzzy-logic based controller in previous proposals. To select appropriate configurations for the controllers and improve the overall performance, the heuristic algorithms CE, PSO and DE are tested in experiments to optimize the control parameters, provided the pre-defined objective function. The simulation suggests that PSO and DE are good candidates as they converge fast. But, in case that time for optimization is not the major concern (e.g. a small network), then CE could be another choice as it can get a lower optimum than PSO and DE, despite at the cost of running longer time.

Apart from the topology control, there are some other ways to attain self-adaptive functionality. Among those techniques, self-adaptive software is a good choice because it provides nice features that allow software configuration at runtime without recompiling or restart applications. For example, components such as reasoner and observer, can be replaced at runtime if alternatives are available. Part of the design of the thesis is eventually implemented and integrated using software engineering, as part of contributions for the DEMANES project. More specifically, the proposal is implemented in the Sun SPOT sensor network and tested in indoor and outdoor scenarios. The node failure is introduced on purpose to evaluate the fault-tolerant capability of the network and the system stability. It turns out that the experimental results match the theories and simulation results. The test results imply that the network is able to offer self-adaptive capability when node failures happen. The network can maintain certain levels of connectivity and stability when few nodes fail, which is the main objective of the thesis.

Due to the network dynamic and unpredictable behaviors, together with resource-constrained feature of nodes, creating a resilient IoT in response to changing environments is not an easy task. The most important lessons learned are:

- The control theory is indeed capable of reaching the design goal in the distributed networks, but at the cost of taking a long time to reach stability, and perhaps requiring lots of computational resources and energy.
- The ways to model the IoT are very complicated, but also very important, because many works rely on its modeling accuracy.
- Most of the work has been done with the help of computer simulations, particularly the optimization part. Because of the iterative nature of the optimization algorithms, without a relatively practical simulation environment and powerful simulation tools, it is very unlikely to have a network with thousands of nodes optimized.
- Self-adaptive software engineering is a promising field that can promote the robustness and the flexibility of applications in the IoT.

7.2 Future works

This section discusses several challenging issues that are expected to be considered in future research. On the one hand, the models and evaluation methods that have been used in this work can be improved in different ways. On the other hand, there are many models that have not been used in current work would be worth trying in the near future.

7.2.1 Improve models and evaluation methods

(1) *Consider other node distributions*

It would be necessary to think about other distribution models when deploying nodes. The mathematical models of this thesis justify that network resilience can be improved by adjusting the communication range (or transmission power) of nodes. But it relies on the conditions that the number of nodes in a network should be big enough, and the nodes are randomly and uniformly placed. It is evident that both conditions are hard to be satisfied in real networks. The experiments carefully put the sensor nodes in the field, and make those nodes uniformly distributed. For instance, this thesis tried to avoid that some nodes form a cluster too far away from the rest of nodes. In turn, the results have somehow supported the theory of this thesis. However, it is likely to fail the network connectivity test if the nodes are inappropriately placed in the field. Therefore, considering other node distributions than an uniform one would be necessary in the future works.

(2) *Experiment with more nodes*

More sensor nodes could be deployed in the field. Due to the limited number of nodes in the lab, the experiments have been performed with only the maximum of 11 sensor nodes. More nodes than 11 might enrich the observations.

(3) *Evaluate real energy consumption*

The energy consumption, which is due to the change of power transmission, is difficult to measure in a sensor node. There are many reasons that can cause different energy consumption among nodes, for instance the diversity of hardware platforms (computing and storage resources), or different applications/tasks running in sensors. Therefore, the energy consumption-related problems are only evaluated in the simulation environment in the current work. Hence, finding a way to evaluate the energy consumption in a real application would be needed.

(4) *Optimize real deployment*

The controller optimization has been evaluated exclusively in the simulator, because the optimization process requires tens of iterations, which is not applicable in a real network. Besides, each iteration needs to be performed under exactly the same initial conditions, e.g. the same amount of battery for each iteration, which is infeasible in practice. Thus, optimizing the system in a real world scenario would be challenging but necessary.

(5) *Complete tests for other proposals*

Not all of the designs are implemented in real sensor nodes, e.g. the proposals in Chapter 4, because they are designed for sensors with richer processing and memory resources, but are implemented and evaluated in the simulator. As our future work, the real world tests for all of the proposals would enrich the work of this thesis.

7.2.2 Extend current models

(1) *More realistic transmission model for IoT*

The network model in many research papers is still too ideal, so it becomes impractical to be tested. As shown in [55][45] for the TelosB testbed and [93][139] for the MicaZ testbed, the empirical studies show that link quality varies significantly over time due to the human activity and multi-path effect. [55] claims that increasing transmission power up to -7dBm can improve the link quality; however, by further increasing the transmission power, the link quality actually decreases. For specific quantities of transmission power, communication distance, and antenna direction, the link quality still varies over time [93]. Those practical results imply that the realistic model of the network is much more complex, and the probabilistic approach can play an important role here. [58] provides an alternative model for an irregular communication range. The irregular radius is modeled as degree of irregularity (DOI), which is defined as the maximum radius range variation per unit degree change in the direction of radio propagation. There are upper and lower bounds for radio propagation. If the neighbor node is above the upper bound, then it is also out of the communication range; while if the node is below the lower bound, it is in the communication range; if the distance is between upper and lower bound, the model is described as DOI. In other words, the DOI parameter can be changed to approximate a specific hardware (e.g. MicaZ) or environment (e.g. indoor). The latest research proposes to use the Cox process to model the irregularity of the communication range [66].

In short, the transmission models (such as the disk model), sometimes are inappropriate to model the IoT nodes. The probabilistic approach can be introduced to create more accurate transmission models.

(2) *Sink node*

Most researches study connectivity issues assuming that all nodes in the network are equally important, without considering the roles of the sink or the base station node which are the data center that all nodes' data usually take as final destination. To some extent, if the sink node loses connection with others, the rest of nodes in the network could be useless. Therefore, a network model that considers the role of the sink node is necessary.

(3) *Higher dimension model*

A commonly used model for IoT is of $2-dimension$. However, the IoT can apply also $3-dimension$, such as underwater, atmospheric, high buildings with many floors, mountains, or space communications. Those models, especially referring to transmission models, are necessary to check their performances in $3-dimension$. Very few researches consider resilient IoT issues in $3-dimension$. The conclusion in [103] can be applied from $2-dimension$ to $3-dimension$ when $k<3$. The transmission range must be at least 1.7889 times the sensing range in order to maintain connectivity among nodes for the truncated octahedral deployment, while the transmission range is between 1.4142 and 1.7889 times the sensing range for the hexagonal prism placement strategy or the rhombic dodecahedron placement [5]. [14] proposes how to construct the full coverage and $k-connectivity$ network in $3-dimension$, where $k \leq 4$. Nevertheless, more proposals on higher dimensions are still imperative. A more complete survey on the topology control based on $3-dimension$ model can be found in [10].

(4) *Mobility*

IoT could be a highly dynamical network where the nodes move frequently; for example, it can be applied to vehicle monitoring or wearable sensor devices. It is interesting to study the network connectivity in a dynamical network. It is very likely that those problems, intractable in a static network, could not be easily solved in a dynamic network either. Although connectivity is more difficult to model and maintain in a mobile environment, the network resilience may benefit from nodes mobility. For example, a careful control of the node trajectory may enhance connectivity and the IoT can recover from failures more quickly.

(5) *Distributed algorithms*

As shown in Table 2.1, many topology control approaches are centralized and have unaffordable complexity. The IoT is a large distributed system, so heuristic algorithms have to be distributed as well. Those non-distributed algorithms harm the scalability and require a higher maintenance cost. The proposed algorithms in this thesis are distributed, but their performances in the distributed networks need to be further studied and improved.

Bibliography

[1] Nedal Ababneh. (2009). Radio irregularity problem in wireless sensor networks: New experimental results. In *Sarnoff Symposium*, pages 1–5. IEEE.

[2] M. Abellanas, A. García, F. Hurtado, J. Tejel, and J. Urrutia. (2008). Augmenting the connectivity of geometric graphs. *Computational Geometry*, 40(3):220–230.

[3] Bondy Adrian and U.S.R. Murty. (2008). *Graph Theory*. Springer.

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

[5] S. M. Nazrul Alam and Zygmunt J. Haas. (2006). Coverage and connectivity in three-dimensional networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 346–357, New York, NY, USA. ACM.

[6] Réka Albert, Hawoong Jeong, and Albert-László Barabási. (2000). Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382.

[7] Abdullah Alfadhly, Uthman Baroudi, and Mohamed Younis. (2012). An effective approach for tolerating simultaneous failures in wireless sensor and actor networks. In *Proceedings of the first ACM international workshop on Mission-oriented wireless sensor networking*, MiSeNet '12, pages 21–26, New York, NY, USA. ACM.

[8] Eiman Alotaibi and Biswanath Mukherjee. (2012). A survey on routing algorithms for wireless ad-hoc and mesh networks. *Computer Networks*, 56(2):940–965.

[9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787–2805.

[10] Azrina Abd Aziz, Y. Ahmet Sekercioglu, Paul Fitzpatrick, and Milosh Ivanovich. (2013). A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks. *IEEE Communications Surveys Tutorials*, 15(1):121–144.

[11] Hakan Bagci and Adnan Yazici. (2013). An energy aware fuzzy approach to unequal clustering in wireless sensor networks. *Applied Soft Computing*, 13(4):1741 – 1749.

[12] Xiaole Bai, Santosh Kumar, Dong Xuan, Ziqiu Yun, and Ten H. Lai. (2006). Deploying wireless sensors to achieve both coverage and connectivity. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 131–142, New York, NY, USA. ACM.

[13] Xiaole Bai, Dong Xuan, Ziqiu Yun, Ten H. Lai, and Weijia Jia. (2008). Complete optimal deployment patterns for full-coverage and k-connectivity ($k \leq 6$) wireless sensor networks. In *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '08, pages 401–410, New York, NY, USA. ACM.

[14] Xiaole Bai, Chuanlin Zhang, Dong Xuan, Jin Teng, and Weijia Jia. (2009). Low-connectivity and full-coverage three dimensional wireless sensor networks. In *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '09, pages 145–154, New York, NY, USA. ACM.

[15] A. Balamurugan and T. Purusothaman. (2012). Ipsd: new coverage preserving and connectivity maintenance scheme for improving lifetime of wireless sensor networks. *WTOC*, 11(1):26–36.

[16] Paul Balister, Béla Bollobás, Amites Sarkar, and Mark Walters. (2009). Highly connected random geometric graphs. *Discrete Applied Mathematics*, 157(2):309–320.

[17] Paul Balister, Amites Sarkar, and Béla Bollobás. (2008). *Percolation, Connectivity, Coverage and Colouring of Random Geometric Graphs*, volume 18 of *Bolyai Society Mathematical Studies*. Springer, Berlin Heidelberg.

[18] Ataul Bari, Arunita Jaekel, Jin Jiang, and Yufei Xu. (2012). Design of fault tolerant wireless sensor networks satisfying survivability and lifetime requirements. *Computer Communications*, 35(3):320–333.

[19] Christian Bettstetter. (2002). On the minimum node degree and connectivity of a wireless multihop network. In *MobiHoc '02 Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, pages 80–91, New York, NY, USA. ACM.

[20] Hao Bin, Tang Han, and Xue Guoliang. (2004). Fault-tolerant relay node placement in wireless sensor networks: formulation and approximation. In

Workshop on High Performance Switching and Routing (HPSR), pages 246–250. IEEE.

[21] Matt Bishop, Marco Carvalho, Richard Ford, and Liam M. Mayron. (2011). Resilience is more than availability. In *Proceedings of the 2011 workshop on New security paradigms workshop*, NSPW '11, pages 95–104, New York, NY, USA. ACM.

[22] Douglas M. Blough, Mauro Leoncini, Giovanni Resta, and Paolo Santi. (2006). The k-neighbors approach to interference bounded and symmetric topology control in ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(9):1267–1282.

[23] Jonathan L. Bredin, Erik D. Demaine, MohammadTaghi Hajiaghayi, and Daniela Rus. (2005). Deploying sensor networks with guaranteed capacity and fault tolerance. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '05, pages 309–319, New York, NY, USA. ACM.

[24] M. R. Brito, E. L. Chávez, A. J. Quiroz, and J. E. Yukich. (1997). Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. *Statistics & Probability Letters*, 35(1):33–42.

[25] Andrei Broder, Michael Fischer, Danny Dolev, and Barbara Simons. (1984). Efficient fault tolerant routings in networks. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC '84, pages 536–541, New York, NY, USA. ACM.

[26] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. (2010). An improved lp-based approximation for steiner tree. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 583–592, New York, NY, USA. ACM.

[27] Haiyan Cai, Xiaohua Jia, and Mo Sha. (2011). Critical sensor density for partial connectivity in large area wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 7(4):1–23.

[28] Duncan S. Callaway, M. E. Newman, Steven H. Strogatz, and Duncan J. Watts. (2000). Network robustness and fragility: percolation on random graphs. *Phys Rev Lett*, 85(25):5468–5471.

[29] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. (2002). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–494.

[30] Xian-Yi Chen and Zhi-Gang Jin. (2012). Research on key technology and applications for internet of things. *Physics Procedia*, 33(0):561 – 566.

[31] Xiuzhen Cheng, Ding-Zhu Du, Lusheng Wang, and Baogang Xu. (2008). Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3):347–355.

[32] Shen Chien-Chung, C. Srisathapornphat, Liu Rui, Huang Zhuochuan, Chaiporn Jaikaeo, and Errol L. Lloyd. (2004). Cltc: a cluster-based topology control for ad hoc networks. *IEEE Transactions on Mobile Computing*, 3(1):18–32.

[33] Kenjiro Cho, Cristel Pelsser, Randy Bush, and Youngjoon Won. (2011). The japan earthquake: the impact on traffic and routing observed by a local isp. In *Proceedings of the Special Workshop on Internet and Disasters*, SWID '11, pages 1–8, New York, NY, USA. ACM.

[34] Liu Chong, Wu Kui, Xiao Yang, and Sun Bo. (2006). Random coverage with guaranteed connectivity: joint scheduling for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(6):562–575.

[35] Paolo Costa, Matteo Cesana, Stefano Brambilla, and Luca Casartelli. (2009). A cooperative approach for topology control in wireless sensor networks. *Pervasive and Mobile Computing*, 5(5):526 – 541.

[36] Paolo Crucitti, Vito Latora, Massimo Marchiori, and Andrea Rapisarda. (2004). Error and attack tolerance of complex networks. *Physica a-Statistical Mechanics and Its Applications*, 340(1-3):388–394.

[37] Robert D. Poor. (2000). Gradient routing in ad hoc networks. Technical report, Massachusetts Institute of Technology.

[38] Deepak R. Dandekar and P. R. Deshmukh. (2012). Relay node placement for multi-path connectivity in heterogeneous wireless sensor networks. *Procedia Technology*, 4(0):732–736.

[39] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67.

[40] Bastian Degener, Sándor P. Fekete, Barbara Kempkes, and Friedhelm Meyer auf der Heide. (2011). A survey on relay placement with runtime and approximation guarantees. *Computer Science Review*, 5(1):57–68.

[41] DEMANES. Design, monitoring and operation of adaptive networked embedded systems. http://www.demanes.eu/, Accessed January 24, 2015.

[42] Min Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. (2005). Localized fault-tolerant event boundary detection in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 902–913. IEEE.

[43] Min Ding, Fang Liu, Andrew Thaeler, Dechang Chen, and Xiuzhen Cheng. (2007). Fault-tolerant target localization in sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2007(1):1–8.

[44] Devdatt Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. (2005). Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *Journal of Computer and System Sciences*, 71(4):467–479.

[45] Russell C. Eberhart and Yuhui Shi. (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 1, pages 81–86.

[46] David Fotue, Foued Melakessou, and Thomas Engel. (2010). Design of an enhanced energy conserving routing protocol based on route diversity in wireless sensor networks. In *The 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 1–7.

[47] Michael L. Fredman and Robert Endre Tarjan. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615.

[48] Zwe-Lee Gaing. (2004). A particle swarm optimization approach for optimum design of pid controller in avr system. *IEEE Transactions on Energy Conversion*, 19(2):384–391.

[49] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. (2001). Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25.

[50] Amitabha Ghosh and Sajal K. Das. (2008). Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive and Mobile Computing*, 4(3):303–334.

[51] Andrew V. Goldberg and Robert E. Tarjan. (1988). A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940.

[52] Francesca Guerriero, Antonio Violi, Enrico Natalizio, Valeria Loscri, and Carmelo Costanzo. (2011). Modelling and solving optimal placement problems in wireless sensor networks. *Applied Mathematical Modelling*, 35(1):230–241.

[53] Sudipto Guha and Samir Khuller. (1998). Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387.

[54] M. Ammari Habib and K. Das Sajal. (2009). Fault tolerance measures for large-scale wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems*, 4(1):1–28.

[55] Gregory Hackmann, Octav Chipara, and Chenyang Lu. (2008). Robust topology control for indoor wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 57–70, New York, NY, USA. ACM.

[56] Mohammad Taghi Hajiaghayi, Nicole Immorlica, and Vahab S. Mirrokni. (2007). Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks. *IEEE/ACM Transactions on Networking*, 15(6):1345–1358.

[57] Frank Harary. (1962). The maximum connectivity of a graph. *Proc Natl Acad Sci U S A.*, 48(7):1142–1146.

[58] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. (2003). Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, pages 81–95, New York, NY, USA. ACM.

[59] R. Hekmat and P. Van Mieghem. (2003). Degree distribution and hopcount in wireless ad-hoc networks. In *Networks, 2003. ICON2003. The 11th IEEE International Conference on*, pages 603–609. IEEE.

[60] Remco Van Der Hofstad, (2010). Random graphs and complex networks. http://www.win.tue.nl/ rhofstad/NotesRGCN2010.pdf, Accessed Feb. 24, 2015.

[61] Haigen Hu, Lihong Xu, Ruihua Wei, and Bingkun Zhu. (2011). Multi-objective control optimization for greenhouse environment using evolutionary algorithms. *Sensors*, 11(6):5792–5807.

[62] Chi-Fu Huang, Yu-Chee Tseng, and Hsiao-Lu Wu. (2007). Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Transactions on Sensor Networks (TOSN)*, 3(1):5.

[63] Yuanjiang Huang, Raúl M. del Toro, José-Fernán Martínez, Vicente Hernández, and Rodolfo Haber. (2014). Connectivity control in wsn based on fuzzy logic control. *ACM SIGBED Review*, 11(3):54–57.

[64] Yuanjiang Huang, José-Fernán Martínez, Vicente Hernández Díaz, and Juana Sendra. (2014). Localized and energy-efficient topology control in wireless sensor networks using fuzzy-logic control approaches. *Mathematical Problems in Engineering*, 2014:1–11.

[65] Yuanjiang Huang, José-Fernán Martínez, Vicente Hernández Díaz, and Juana Sendra. (2014). A novel topology control approach to maintain the node degree in dynamic wireless sensor networks. *Sensors*, 14(3):4672–4688.

[66] Yuanjiang Huang, José Fernán Martínez Ortega, Juana Sendra, and Lourdes López Santidrián. (2013). The Influence of communication range on connectivity for resilient wireless sensor networks using a probabilistic approach. *International Journal of Distributed Sensor Networks*, 2013:1 – 11.

[67] Ming-Hao Hung, Li-Sun Shu, Shinn-Jang Ho, Shiow-Fen Hwang, and Shinn-Ying Ho. (2008). A novel intelligent multiobjective simulated annealing algorithm for designing robust pid controllers. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38(2):319–330.

[68] Bill Jackson and Tibor Jordán. (2005). Independence free graphs and vertex connectivity augmentation. *Journal of Combinatorial Theory, Series B*, 94(1):31–77.

[69] David B. Johnson and David A. Maltz. (1996). Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, pages 152–181.

[70] David R. Karger, Philip N. Klein, and Robert E. Tarjan. (1995). A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328.

[71] Gaurav S. Kasbekar, Yigal Bejerano, and Saswati Sarkar. (2012). Generic coverage verification without location information using dimension reduction. *IEEE/ACM Transactions on Networking*, 20(6):1991–2004.

[72] Abhishek Kashyap, Samir Khuller, and Mark Shayman. (2006). Relay placement for higher order connectivity in wireless sensor networks. In *INFOCOM Proceedings of 25th IEEE International Conference on Computer Communications*, pages 1–12. IEEE.

[73] Tierney Kathleen and Bruneau. Michel. (2007). Conceptualizing and measuring resilience: A key to disaster loss reduction. *TR NEWS*, pages 14–17.

[74] C. Kavitha and K. V. Viswanatha. (2009). An energy efficient fault tolerant multipath (eeftm) routing protocol for wireless sensor networks. In *IEEE International Advance Computing Conference*, pages 746–751. IEEE.

[75] Tohru Kawabe and Takanori Tagami. (1997). A real coded genetic algorithm for matrix inequality design approach of robust pid controller with two degrees of freedom. In *Proceedings of the 1997 IEEE International Symposium on Intelligent Control*, pages 119–124. IEEE.

[76] M. Passino Kevin and Yurkovich Stephen. (1998). *Fuzzy Control*. Addison Wesley Longman, Inc.

[77] Maleq Khan and Gopal Pandurangan. (2008). A fast distributed approximation algorithm for minimum spanning trees. *Distributed Computing*, 20(6):391–402.

[78] Samir Khuller and Uzi Vishkin. (1994). Biconnectivity approximations and graph carvings. *Journal of the ACM (JACM)*, 41(2):214–235.

[79] Santosh Kumar, Ten H. Lai, and József Balogh. (2004). On k-coverage in a mostly sleeping sensor network. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, MobiCom '04, pages 144–158, New York, NY, USA. ACM.

[80] Robert Laddaga. (1997). Self-adaptive software. Technical report, MIT Artificial Intelligence Laboratory.

[81] Robert Laddaga. (2000). Active software. In *Proceedings of the First International Workshop on Self-adaptive Software*, IWSAS' 2000, pages 11–26, Secaucus, NJ, USA. Springer-Verlag New York, Inc.

[82] Chuen Chien Lee. (1990). Fuzzy logic in control systems: fuzzy logic controller. i. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418.

[83] Myeong-Hyeon Lee and Yoon-Hwa Choi. (2008). Fault detection of wireless sensor networks. *Computer Communications*, 31(14):3469–3475.

[84] Sookyoung Lee and Mohamed Younis. (2010). Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree. *Journal of Parallel and Distributed Computing*, 70(5):525–536.

[85] Yong Oh Lee and A. L. Narasimha Reddy. (2012). Constructing disjoint paths for failure recovery and multipath routing. *Computer Networks*, 56(2):719–730.

[86] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. (1990). The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105–115.

[87] Li Li, Joseph Y. Halpern, Paramvir Bahl, Yi-Min Wang, and Rogert Wattenhofer. (2001). Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks. In *Proceedings of the Twentieth Annual ACM*

Symposium on Principles of Distributed Computing, PODC '01, pages 264–273, New York, NY, USA. ACM.

[88] Ning Li and Jennifer C. Hou. (2006). Localized fault-tolerant topology control in wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(4):307–320.

[89] Ning Li, Jennifer C. Hou, and Lui Sha. (2003). Design and analysis of an mst-based topology control algorithm. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1702–1712. IEEE.

[90] Xiang-Yang Li, Wen-Zhan Song, and Yu Wang. (2005). Efficient topology control for ad-hoc wireless networks with non-uniform transmission ranges. *Wireless Networks*, 11(3):255–264.

[91] Xuefei Li and Laurie Cuthbert. (2004). A reliable node-disjoint multipath routing with low overhead in wireless ad hoc networks. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '04, pages 230–233, New York, NY, USA. ACM.

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

[93] Shan Lin, Jingbin Zhang, Gang Zhou, Lin Gu, John A. Stankovic, and Tian He. (2006). Atpc: adaptive transmission power control for wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 223–236, New York, NY, USA. ACM.

[94] Yanheng Liu, An Ren, Dayang Sun, and Aimin Wang. (2013). A proactive maintaining algorithm for dynamic topology control in wireless sensor networks. *Computers & Electrical Engineering*, 39(6):1767 – 1778.

[95] Dmitri Loguinov, Anuj Kumar, Vivek Rai, and Sai Ganesh. (2003). Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 395–406, New York, NY, USA. ACM.

[96] Xuanwen Luo, Ming Dong, and Yinlun Huang. (2006). On distributed fault-tolerant detection in wireless sensor networks. *IEEE Transactions on Computers*, 55(1):58–70.

[97] Renita Machado and Sirin Tekinay. (2008). A survey of game-theoretic approaches in wireless sensor networks. *Computer Networks*, 52(16):3047–3061.

[98] Clémence Magnien, Matthieu Latapy, and Jean-Loup Guillaume. (2011). Impact of random failures and attacks on poisson and power-law random networks. *ACM Computing Surveys (CSUR)*, 43(3):1–31.

[99] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, and M.F. Tasgetiren. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11(2):1679–1696.

[100] MathWorks. Matlab. http://www.mathworks.com/, Accessed January 24, 2015.

[101] Yu Mengjie, Hala Mokhtar, and Majid Merabti. (2007). Fault management in wireless sensor networks. *IEEE Wireless Communications*, 14(6):13–19.

[102] Kenji Miyao, Hidehisa Nakayama, Nirwan Ansari, and Nei Kato. (2009). Ltrt: An efficient and reliable topology control algorithm for ad-hoc networks. *IEEE Transactions on Wireless Communications*, 8(12):6050–6058.

[103] Bahramgiri Mohsen, Hajiaghayi Mohammadtaghi, and S. Mirrokni Vahab. (2006). Fault-tolerant and 3-dimensional distributed topology control algorithms in wireless multi-hop networks. *Wireless Networks*, 12(2):179–188.

[104] Walid Najjar and Jean-Luc Gaudiot. (1990). Network resilience: a measure of network fault tolerance. *IEEE Transactions on Computers*, 39(2):174–181.

[105] J. Ni and S. A. G. Chandler. (1994). Connectivity properties of a random radio network. *IEE Proceedings-Communications*, 141(4):289–296.

[106] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. (2009). Acm transactions on sensor networks (tosn). *ACM Trans. Sen. Netw.*, 5(3):1–29.

[107] Ashutosh Nigam and Yogesh K. Agarwal. (2014). Optimal relay node placement in delay constrained wireless sensor network design. *European Journal of Operational Research*, 233(1):220 – 233.

[108] Oracle. Sun spot. http://www.sunspotworld.com/, Accessed October 2, 2014.

[109] OSGi. Open service gateway initiative. http://www.osgi.org/, Accessed January 24, 2015.

[110] Mathew D. Penrose. (1999). On k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 15(2):145–164.

[111] Hassaan Khaliq Qureshi, Sajjad Rizvi, Muhammad Saleem, Syed Ali Khayam, Veselin Rakocevic, and Muttukrishnan Rajarajan. (2011). Poly: A reliable and energy efficient topology control protocol for wireless sensor networks. *Computer Communications*, 34(10):1235 – 1242.

[112] Marjan Radi, Behnam Dezfouli, Shukor Abd Razak, and Kamalrulnizam Abu Bakar. (2010). Liemro: A low-interference energy-efficient multipath routing protocol for improving qos in event-based wireless sensor networks. In *Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pages 551–557, Venice, Italy. IEEE.

[113] Melamed Roie, Keidar Idit, and Barel Yoav. (2008). Octopus: a fault-tolerant and efficient ad-hoc routing protocol. *Wireless Networks*, 14(6):777–793.

[114] Charles E. Perkins Royer and Elizabeth M. Royer. (1999). Ad-hoc on-demand distance vector routing. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100.

[115] Indranil Saha, Lokesh Kumar Sambasivan, Subhas Kumar Ghosh, and Ranjeet Kumar Patro. (2010). Distributed fault-tolerant topology control in wireless multi-hop networks. *Wireless Networks*, 16(6):1511–1524.

[116] Pitipatana Sakarindr and Nirwan Ansari. (2007). Security services in group communications over wireless infrastructure, mobile ad hoc, and wireless sensor networks. *IEEE Wireless Communications*, 14(5):8–20.

[117] Mazeiar Salehie and Ladan Tahvildari. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):14:1–14:42.

[118] Fatih Senel and Mohamed Younis. (2011). Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation. *Computer Communications*, 34(16):1932–1941.

[119] Izzet F. Senturk, Kemal Akkaya, and Sabri Yilmaz. (2014). Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information. *Ad Hoc Networks*, 13, Part B(0):487 – 503.

[120] Qingya She, Xiaodong Huang, and Jason P. Jue. (2010). How reliable can two-path protection be? *IEEE/ACM Transactions on Networking*, 18(3):922–933.

[121] Yuhui Shi and Russell C. Eberhart. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation*, volume 3, pages 1945–1950.

[122] Hanan Shpungin and Michael Segal. (2009). Low-energy fault-tolerant bounded-hop broadcast in wireless networks. *IEEE/ACM Transaction on Networking*, 17(2):582–590.

[123] Hanan Shpungin and Michael Segal. (2010). On minimizing the total power of k-strongly connected wireless networks. *Wireless Networks*, 16(4):1075–1089.

[124] Deepinder Sidhu, Raj Nair, and Shukri Abdallah. (1991). Finding disjoint paths in networks. *ACM SIGCOMM Computer Communication Review*, 21(4):43–51.

[125] Anand Srinivas and Eytan Modiano. (2005). Finding minimum energy disjoint paths in wireless ad-hoc networks. *Wireless Networks*, 11(4):401–417.

[126] James P. G. Sterbenz, David Hutchison, Egemen K. Çetinkaya, Abdul Jabbar, Justin P. Rohrer, Marcus Schöller, and Paul Smith. (2010). Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8):1245–1265.

[127] J. W. Suurballe. (1974). Disjoint paths in a network. *Networks*, 4(2):125–145.

[128] Bernd Thallner, Heinrich Moser, and Ulrich Schmid. (2010). Topology control for fault-tolerant communication in wireless ad hoc networks. *Wireless Networks*, 16(2):387–404.

[129] Javad Akbari Torkestani. (2013). An energy-efficient topology construction algorithm for wireless sensor networks. *Computer Networks*, 57(7):1714 – 1725.

[130] Arnold B. Urken, A. Nimz, and Tod M. Schuck. (2012). Designing evolvable systems in a framework of robust, resilient and sustainable engineering analysis. *Advanced Engineering Informatics*, 26(3):553–562.

[131] Vermesan Vidiu, Friess Peter, Guillemin Patrick, Sundmaeker Harald, Eisenhauer Markus, Moessner Klaus, Le Gall Franck, and Cousin Philippe. (2013). Internet of things strategic research and innovation agenda. Technical report, IoT European Research Cluster.

[132] Fabio R.J. Vieira, Jose F. de Rezende, Valmir C. Barbosa, and Serge Fdida. (2013). Local heuristic for the refinement of multi-path routing in wireless mesh networks. *Computer Networks*, 57(1):273 – 285.

[133] Bang Wang. (2011). Coverage problems in sensor networks: A survey. *ACM Computing Surveys (CSUR)*, 43(4):1–53.

[134] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. (2003). Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of the 1st international conference on*

Embedded networked sensor systems, SenSys '03, pages 28–39, New York, NY, USA. ACM.

[135] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. (2010). R3: resilient routing reconfiguration. *ACM SIGCOMM Computer Communication Review*, 41(4):291–302.

[136] Guy E. Weichenberg, Vincent W. S. Chan, and Muriel Medard. (2004). High-reliability architectures for networks under stress. In *INFOCOM Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 131–141.

[137] Elias Weingartner, Hendrik vom Lehn, and Klaus Wehrle. (June 2009). A performance comparison of recent network simulators. In *ICC '09. IEEE International Conference on Communications*, pages 1–5.

[138] WIKI. Service-oriented architecture. http://en.wikipedia.org/wiki/Service-oriented_architecture, Accessed Feb. 24, 2015.

[139] Shuo Xiao, Ashay Dhamdhere, Vijay Sivaraman, and Alison Burdett. (2009). Transmission power control in body area sensor networks for healthcare monitoring. *IEEE Journal on Selected Areas in Communications*, 27(1):37–48.

[140] Han Xiaofeng, Cao Xiang, Errol L. Lloyd, and Shen Chien-Chung. (2010). Fault-tolerant relay node placement in heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(5):643–656.

[141] Bai Xiaole, Yun Ziqiu, Xuan Dong, Ten H. Lai, and Jia Weijia. (2010). Optimal patterns for four-connectivity and full coverage in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(3):435–448.

[142] Guoliang Xing, Chenyang Lu, Xiaohua Jia, and Robert Pless. (2013). Localized and configurable topology control in lossy wireless sensor networks. *Ad Hoc Networks*, 11(4):1345 – 1358.

[143] Cheng Xiuzhen, Bhagirath Narahari, Rahul Simha, Maggie Xiaoyan Cheng, and Dan Liu. (2003). Strong minimum energy topology in wireless sensor networks: Np-completeness and heuristics. *IEEE Transactions on Mobile Computing*, 2(3):248–256.

[144] Feng Xue and P. R. Kumar. (2004). The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10(2):169–181.

[145] International Telecommunication Union ITU-T Y.2060. (2012). Next generation networks-frameworks and functional architecture models - overview of the internet of things. Technical report, International Telecommunication Union.

[146] Bashir Yahya and Jalel Ben-Othman. (2009). An energy efficient and qos aware multipath routing protocol for wireless sensor networks. In *IEEE 34th Conference on Local Computer Networks*, pages 93–100.

[147] Bashir Yahya and Jalel Ben-Othman. (2009). Reer: Robust and energy efficient multipath routing protocol for wireless sensor networks. In *IEEE GLOBECOM Global Telecommunications Conference*, pages 1–7.

[148] Bashir Yahya and Jalel Ben-Othman. (2010). Relax: An energy efficient multipath routing protocol for wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6.

[149] Wenjing Yang, Xinyu Yang, Shusen Yang, and Dongxu Yang. (2011). A greedy-based stable multi-path routing protocol in mobile ad hoc networks. *Ad Hoc Networks*, 9(4):662–674.

[150] Her-Terng Yau, Tzu-Hsiang Hung, and Chia-Chun Hsieh. (2012). Bluetooth based chaos synchronization using particle swarm optimization and its applications to image encryption. *Sensors*, 12(6):7468–7484.

[151] Fan Ye, Gary Zhong, Songwu Lu, and Lixia Zhang. (2002). Peas: A robust energy conserving protocol for long-lived sensor networks. *10th Ieee International Conference on Network Protocols, Proceedings*, pages 200–201.

[152] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. (2008). Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330.

[153] Fu Yong, Sha Mo, Gregory Hackmann, and Lu Chenyang. (2012). Practical control of transmission power for wireless sensor networks. In *20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10.

[154] Mohamed Younis and Kemal Akkaya. (2008). Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Netw.*, 6(4):621–655.

[155] Mohamed Younis, Izzet F. Senturk, Kemal Akkaya, Sookyoung Lee, and Fatih Senel. (2014). Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks*, 58(0):254 – 283.

[156] Huang Yuanjiang, Martínez José-Fernán, Sendra Juana, and López Lourdes. (2014). A survey on resilient wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, under the second round review.

[157] Huang Yuanjiang, Martínez José-Fernán, Sendra Juana, and López Lourdes. (2015). Parameter optimization for the control system in wireless sensor networks. *Wireless Networks*, under the second round review.

[158] Huang Yuanjiang, Martínez José-Fernán, Sendra Juana, and Néstor Lucas Martínez. (2015). Design and implementation of control system for resilient wireless sensor networks. *Mobile Networks and Applications*, under review.

[159] Lotfi A. Zadeh. (1965). Fuzzy sets. *Information and Control*, 8(3):338 – 353.

[160] Dongsheng Zhang, Santosh Ajith Gogi, Dan S. Broyles, Egemen K. Çetinkaya, and James P.G. Sterbenz. (2012). Modelling wireless challenges. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 423–426, New York, NY, USA. ACM.

[161] Zhongshan Zhang, Keping Long, Jianping Wang, and Falko Dressler. (2014). On swarm intelligence inspired self-organized networking: Its bionic mechanisms, designing principles and optimization approaches. *IEEE Communications Surveys Tutorials*, 16(1):513–537.

[162] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. (2004). Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 125–138, New York, NY, USA. ACM.

List of Author's Publications

Most of the results of this thesis were published or submitted in the following journals indexed in Journal Citation Reports (JCR) and international conference:

1. **Yuanjiang Huang**, José-Fernán Martínez, Vicente Hernández Díaz, and Juana Sendra. (2014). *A Novel Topology Control Approach to Maintain the Node Degree in Dynamic Wireless Sensor Networks.* Sensors, 14(3): 4672-4688 (**JCR Q1**, **Impact Factor = 2.048**, **T1**), ISSN:1424–8220.
2. **Yuanjiang Huang**, José-Fernán Martínez, Vicente Hernández Díaz, and Juana Sendra. (2014). *Localized and Energy-Efficient Topology Control in Wireless Sensor Networks Using Fuzzy-Logic Control Approaches.* Mathematical Problems in Engineering, 2014: 1-11 (**JCR Q2**, **Impact Factor = 1.082**, **T2**), ISSN:1024–123X.
3. **Yuanjiang Huang**, José-Fernán Martínez, Juana Sendra, and Lourdes López. (2013). *The Influence of Communication Range on Connectivity for Resilient Wireless Sensor Networks Using a Probabilistic Approach.* International Journal of Distributed Sensor Networks, 2013: 1-11 (**JCR Q3**, **Impact Factor = 0.923**, **T2**), ISSN: 1550–1329.
4. **Yuanjiang Huang**, Raúl M. Del Toro, José-Fernán Martínez Ortega, Vicente Hernández Díaz, and Rodolfo Haber. (2014). *Connectivity control in WSN based on Fuzzy Logic Control.* ACM SIGBED Review - Special Issue on the 6th Workshop on Adaptive and Reconfigurable Embedded Systems, 11(3):54-57, Berlin, Germany, DOI=10.1145/2692385.2692395, ISSN:1551–3688.
5. **Yuanjiang Huang**, José-Fernán Martínez, Juana Sendra, and Lourdes López. (2014, under the second round review). *A survey on resilient wireless sensor networks.* Ad Hoc & Sensor Wireless Networks (**JCR Q4**, **Impact Factor = 0.478**, **T3**), ISSN:1551–9899.
6. **Yuanjiang Huang**, José-Fernán Martínez, and Juana Sendra. (2015, under the second round review). *Parameter Optimization for the Control System in Wireless Sensor Networks.* Wireless Networks (**JCR Q3**, **Impact Factor = 1.055**, **T2**), ISSN:1022–0038.
7. **Yuanjiang Huang**, José-Fernán Martínez, Juana Sendra, and Néstor Lucas Martínez. (2015, under review). *Design and Implementation of Control System for Resilient Wireless Sensor Networks.* Mobile Networks and Applications (**JCR Q2**, **Impact Factor = 1.496**, **T2**), ISSN:1383–469X.