Robust and Adaptive Network Architecuture for Internet of Things

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2017

Shinya Toyonaga

List of publication

Journal papers

1. Shinya Toyonaga, Daichi Kominami, Masashi Sugano and Masayuki Murata, "Potential-based Routing for Supporting Robust Any-to-any Communication in Wireless Sensor Networks," *EURASIP Journal of Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–13, December 2013.

2. Shinya Toyonaga, Daichi Kominami and Masayuki Murata, "Virtual Wireless Sensor Networks: Adaptive Brain-Inspired Configuration for Internet of Things Applications," *Sensors*, vol. 16, no. 8, p. 1323, August 2016.

Refereed Conference Papers

1. Shinya Toyonaga, Daichi Kominami, Masashi Sugano and Masayuki Murata, "Potential-based Downstream Routing for Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Systems and Networks Communications (ICSNC 2012)*, pp. 59–64, November 2012.

2. Shinya Toyonaga, Yuki Fujita, Daichi Kominami and Masayuki Murata, "Implementation of Controlled Sink Mobility Strategies with a Gradient Field in Wireless Sensor Networks," in *Proceedings of the 7th International Conference on Sensor Technologies and Applications (SENSORCOMM 2013)*, pp. 25–31, August 2013.

3. Shinya Toyonaga, Daichi Kominami and Masayuki Murata, “Brain-inspired Method for Constructing a Robust Virtual Wireless Sensor Network,” in *Proceedings of the 2015 International Conference on Computing and Network Communications (CoCoNet 2015)*, pp. 59–65, December 2015.

Non-Refereed Technical Papers

1. Shinya Toyonaga, Daichi Kominami, Masashi Sugano, Masayuki Murata and Takaaki Hatauchi, “Evaluation of Potential-based Downstream Routing on Wireless Sensor Networks,” *Technical Report of IEICE (AN2012-5)*, vol. 112, no. 30, pp. 45–46, May 2012 (in Japanese).

2. Shinya Toyonaga, Daichi Kominami and Masayuki Murata, “Method for Constructing a Robust Virtual Sensor Network Inspired by Brain Networks,” *IEICE Technical Committee on Information Network Science (NetSci)*, May 2014 (in Japanese).

Preface

Many researchers have been devoting significant attention to the Internet of Things (IoT) or cyber-physical systems. The research area for IoT architecture extends over all network, such as cloud services on data center networks, software-defined network on core networks, fog computing on edge networks and self-organizing systems on sensor networks. Above all, collecting environmental information is one of the most important part of the IoT for interaction between human and environment. Thus, we need to study wireless sensor networks (WSNs) more deeply to realize the IoT. In the IoT environment, WSNs will be a critical technology for allowing collecting and acting on environmental information for such potential applications as health-care monitoring, emergency responses, accident detection, and tracking.

WSNs, which comprise many sensor/sink nodes, are expected to be integrated into future communication infrastructures. However, WSNs have been an architecture placing its importance on collection of environmental data, such as temperature, humidity, illuminance, oscillation, and movement. In WSNs for data collection, the main problems are comping with an energy-scarce, harsh wireless environment, and managing a large number of devices. In this vein, energy efficiency, robustness, and scalability have been addressed for the past few decades, and methods limited to specific situations have been proposed. As a result, providing stable applications remains difficult, and most existing research has not considered IoT-specific situations.

In the near future, billions of devices are expected to connect to the Internet and to communicate with one another for various applications. Therefore, in IoT environment, it is important to manage and maintain a great number of devices. Largeness of scale gives rise to some problems to be solved, such as scalability and heterogeneity in device. Thus, WSN architecture for IoT must be

designed so as to overcome the issues with scalability and heterogeneity. Reliability, even in the face of environmental changes, is also important. Within the IoT environment, changes in traffic patterns, variations in traffic demand, and addition or removal of sensor nodes can all occur.

Therefore, in this thesis, we provide an architecture for reliable Internet of Things (IoT) applications on WSNs. For this, we define reliability as consisting of robustness and adaptivity. To design a reliable architecture for WSNs, we must consider the physical layer, the MAC layer, the routing layer, and the application layer (including cross-layer optimization) on devices, as well as service provisioning by aggregation of devices for each application. Our key ideas are a self-organizing system to earn adaptivity and scalability, and a virtual topology construction method inspired-by brain network structures to earn robustness and to solve heterogeneity. In this thesis, we investigate the routing layer together with the MAC layer, and propose a virtual topology construction method for WSNs as a method of service provisioning. Then, we evaluate the reliability of the proposed method using packet-level simulation.

We first propose an any-to-any routing protocol possessing scalability and adaptivity against environmental changes. For this purpose, we take potential-based routing having scalability, adaptivity, and energy efficiency, and extend it to any-to-any communication. We show that this approach retains the original advantages of potential-based routing. Originally, potential-based routing was designed for upstream (sensor-to-sink) communication, and since only an upstream communication pattern and limited resources of sensor nodes are assumed, many methods have been proposed. In potential-based upstream routing (PBUR), each node has a scalar value known as "potential,"and calculates its own potential by using the information of its neighboring nodes. PBUR has scalability because each sensor node manages one only scalar value using local communication. We first extend potential-based routing for downstream (sink-to-sensor) communication by using multiple potential fields. The main idea is to identify a virtual position of a sensor node using trilateration. At least three potential fields thus become capable of downstream communication. Then, we achieve a potential-based any-to-any routing protocol (PBAR) by merging potential-based upstream and downstream routing. In PBAR, sensor nodes can send data to a certain node by routing the data via a sink node. Our proposal is shown to be highly adaptive to node failure by way of simulation evaluation.

Next, we consider how to integrate multiple heterogeneous WSNs, and how an application developer could lease reliable sensor resources. In a future IoT environment, all devices would be connected to the Internet, and would communicate with one another to meet the demands of various applications. When we consider WSNs as communication infrastructure of the future IoT, integrating deployed sensor resources will be a critical consideration. Although the concept of virtualization of sensor networks, which is a technique for integrating multiple WSNs, has been advocated, designing virtualized WSNs for actual environments will require further detailed studies. In this study, we show an overview of an architecture that is reliable for constructing and running virtual wireless sensor network (VWSN) services within a VWSN topology that has a hierarchical modular structure. This approach provides users (application developers) with a reliable VWSN network by assigning redundant resources according to each user's demand and providing a recovery method to incorporate environmental changes. We tested this approach by simulation experiment, with results showing that a VWSN is reliable in many cases, although the physical deployment of sensor nodes and the modular structure of the VWSN will be quite important to the stability of services within the VWSN topology.

From the studies mentioned above, we found that a connection pattern between modules is critical for robust connectivity and stability of services. To increase reliability of the VWSN topology mentioned above, we use an analytical approach and propose a method of characterizing the robustness of a topology according to the inter-module embedded links. Our proposal is to investigate percolation dynamics in a modular network, particularly graph ensembles after addition of inter-module links. In considering adding inter-module links, we extend a binary-dynamics model, which is an analytical model for estimating robustness of modular networks. Evaluation by simulation showed that graphs in which inter-module links connect lower-degree nodes with one another, and in which the number of endpoint nodes of inter-module links is high, have robust connectivity. We additionally investigate the applicable range of our proposed method.

Acknowledgments

First of all, I would like to express my great gratitude to my supervisor, Professor Masayuki Murata, for his meaningful advice and valuable discussions throughout my Ph.D.

I am heartily grateful to the members of my thesis committee, Professor Takashi Watanabe, Professor Teruo Higashino and Professor Toru Hasegawa of Graduate School of Information Science and Technology, Osaka University, and Professor Morito Matsuoka of Cyber Media Center, Osaka University, for their multilateral reviews and perceptive comments.

Also, I would like to express my sincere appreciation for Professor Masashi Sugano of School of Knowledge and Information Systems, College of Sustainable System Sciences, Osaka Prefecture University. He is the first person who taught me a delight and difficulty of research. I would like to thank Assistant Professor Daichi Kominami of Graduate School of Economics, Osaka University. Without his continuous advices and supports, I would not have entered the Ph.D. program. He made my daily life fruitful.

Furthermore, I must acknowledge Associate Professor Shin'ichi Arakawa, Associate Professor Go Hasegawa, Assistant Professor Yuichi Ohsita of Graduate School of Information Science and Technology, Osaka University, and Dr. Kenji Leibnitz of National Institute of Information and Communications Technology, for their valuable comments and suggestions on my study.

I express my appreciation to all of past and present colleagues, friends, and secretaries of the Advanced Network Architecture Research Laboratory, Graduate School of Information Science and Technology, Osaka University.

I cannot conclude my acknowledgement without expressing my thanks to my parents and family. Thank you for your giving me invaluable supports throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the near future, billions of devices, including PCs, mobile phones, and sensing or actuation devices, are expected to connect to the Internet and communicate with one another to allow a safe, secure, and comfortable life. In such a large-scale network, the Internet of Things (IoT) attracts a great deal of attention, and many researchers work on the subject [1]. Collecting data for objects or environmental information, analyzing such data, and discovering new values stimulates the service industry. In the field of human welfare, sensors monitor not only temperature and humidity but also human biological conditions and motions to prevent injuries and accidents and to help treatment and rehabilitation [2]. In industry, such networks are used for automation of production or improvement of safety by sensing and actuation [3].

The research area for IoT architecture extends over all network, such as cloud services on data center networks [4], software-defined network on core networks [5], fog computing on edge networks [6] and self-organizing systems on sensor networks [7]. Above all, collecting environmental information is one of the most important part of the IoT for interaction between human and environment. Thus, we need to study wireless sensor networks (WSNs) more deeply to realize the IoT. To achieve an IoT environment, WSNs will be a critical technology for allowing collecting and acting on environmental information.

WSNs, which comprise a large number of sensor/sink nodes, are expected to be integrated into future communication infrastructures [8]. However, WSNs have been an architecture placing its importance on collection of environmental data. In WSNs for data collection, the main problems concern handling an energy-scarce, harsh wireless environment, and managing a high number of devices. In this vein, energy efficiency, robustness, and scalability have been addressed for the past few decades, and methods limited to specific situations, or applied to closed networks, have been proposed [9]. As a result, providing stable applications remains difficult, and most existing research has not considered IoT-specific situations.

The main challenge in IoT is to manage and maintain a great number of devices. Largeness of scale gives rise to some problems to be solved, such as scalability and heterogeneity [1]. Scalability, in particular, is a challenging issue because the amount of data and control packets for managing networks that are produced increases as the number of deployed devices grows. Such heavy traffic leads to packet loss due to congestion and packet collisions. When this occurs, data or services cannot be provided. To deal with the issues of scalability, decision-making with only local information exchanges [9, 10], a hierarchical scheme in which a task is split into smaller subtasks [1], and an abstraction scheme in which networks are divided into modules or clusters [11], have been proposed. Heterogeneity arises from the deployment of various types of devices by multiple vendors. These devices have different energy capacities, computing power, functionalities, and operating conditions. Thus, integration of such devices to solve a certain task is important. Apart from the use of standardized or common protocols, gateway-based and virtualization-based solutions for heterogeneity have been proposed [12, 13]. In addition, the architecture for IoT must be designed so as to overcome the issues with scalability and heterogeneity. Reliability, even in the face of environmental changes, is also important. Within the IoT environment, changes to traffic patterns, variations in traffic demand, and addition or removal of sensor nodes can all occur.

Therefore, in this thesis, we provide an architecture for reliable IoT applications. We define reliability as consisting of robustness and adaptivity. Robustness indicates that performance does not degrade and that services running on the architecture continue to function even when serious disruptions occur. Adaptivity is the property of recovering performance after environmental changes occur and performance degrades temporarily. To design a reliable architecture for WSNs, we must

consider the physical layer, the MAC layer, the routing layer, and the application layer (including cross-layer optimization) on devices, as well as service provisioning by aggregation of devices for each application. Our key ideas are a self-organizing system to earn adaptivity and scalability, and a virtual topology construction method inspired-by brain network structures to earn robustness and to solve heterogeneity. This thesis investigates the routing layer together with the MAC layer and proposes a virtual topology construction method for WSNs as a way of service provisioning. Then, we evaluate the reliability of our proposed method using packet-level simulation.

Routing Protocols

In most applications, sensor nodes are supposed to operate in an unattended manner for a lengthy period after deployment. Due to the harsh environment of WSNs and the high number of sensor nodes, it is not uncommon that sensor nodes become faulty and unreliable [14]. Therefore, it is essential that protocols used in WSNs are reliable. Moreover, energy savings are important in wireless sensor networks because sensor nodes are typically battery-powered.

During the past few decades, various many-to-one upstream (sensor-to-sink) routing protocols have been studied, for which an upstream communication pattern and limited resources of sensors are assumed. However, there is also a demand for downstream (sink-to-sensor) routing [15]. A sink node may send a query to a specific sensor node upon receiving abnormal data from the sensor node, or a sink node may send a message to change the frequency of sensing in a specific domain. Further expansion of future applications of wireless sensor networks will diversify the demands of the networks. To meet the requirements of future WSNs, a sophisticated any-to-any routing protocol must be realized. In applications using this protocol, in-network processing, such as reactive tasking, data querying, or data-centric storage, necessitates communication between sensor nodes [16]. In addition to the typical demands of WSNs, an any-to-any routing protocol must also achieve low energy consumption, high scalability, and strong reliability [17].

For ad hoc WSNs, various any-to-any routing protocols have been studied. In the flooding method and the gossiping method, messages are relayed via broadcasts [18, 19]. These methods suffer from a high number of redundant transmissions, particularly when a few nodes in a specific

domain are destinations. Many studies have been conducted on reactive and proactive routing protocols [20, 21]. In reactive protocols, each node constructs routes only when communication is required. Power consumption can then be reduced when communication is not needed. Delay time, however, is longer for reactive protocols because of route discovery procedures. This means that reactive protocols are not appropriate for real-time applications. In proactive protocols, end-to-end delay is small. However, there is overhead because all nodes collect information about links. Geographic routing protocols allow for communication between two arbitrary nodes [22]. Equipment for acquiring precise geographic positions is required for these protocols, and all nodes must know the positions of their destinations. The virtual coordinate assignment protocol (VCap) is able to route data using a virtual position without the need for a Global Positioning System device [23]. In VCap, all nodes store the three lowest hop counts to three anchor nodes and use them as virtual coordinates. Because hop counts are integers, some nodes may have the same virtual coordinates in VCap. Other virtual coordinate assignment protocols have been proposed [24–28]. Further, many potential-based routing protocols (one type of proactive routing protocol) aim for low overhead, high scalability, and energy balancing [9, 10, 29, 30]. Originally, potential-based routing was designed for upstream communication, and many methods of potential-based routing have been proposed. In potential-based upstream routing (PBUR), each node has a scalar value known as "potential," and calculates its own potential by using the information of its neighboring nodes. PBUR is scalable because each sensor node manages only one scalar value, by local communication. Moreover, load balancing and extension of the lifetime of WSNs using residual energy of neighbor nodes or the amount of traffic have been studied [9, 10, 31]. Because of these advantages, PBUR can be considered as a suitable routing protocol for WSNs. Therefore, we take PBUR and extend it to any-to-any communication to exploit its advantages.

Virtualization of WSNs

For the future IoT environment, effective integration of various types of networks will be an important consideration. Domains may include, in addition to co-existing wired and wireless links, sensor nodes and actuator nodes from multiple vendors. Moreover, applications running over such

mixed networks may make different demands on the network, such as locational information or other node-specific information.

To achieve the IoT environment, WSNs will be a critical technology to allow collecting and acting on environmental information. WSNs will be integrated into future communication infrastructure [8]. Here, however, networks are considered to be constructed independently to provide service within a local area. For such heterogeneous WSNs to be consolidated into infrastructure, and to share physical sensor substrates across multiple IoT applications, virtualization of WSNs is one solution. In virtualization methods, the functionality of a WSN is split into two parts: the physical infrastructure of sensors, and the applications, which rely on the aggregated resources. The main advantage of virtualizing WSNs is that this provides the ability to achieve a shared infrastructure that can satisfy various service demands [32, 33].

For example, sensor substrates deployed for fire-detection or fire-tracking applications can be shared between homeowners and city administrations [34]. In extant task-oriented WSNs, redundant sensor nodes are deployed in a shared domain, and duplicated nodes are chosen according to the application. This is done because the required granularities or service demands of distinct users are different, but redundant deployment would be inefficient. Virtualization can eliminate tight coupling between applications and resources, thus enabling the use of existing sensor substrates in multiple applications. Moreover, virtualization of WSNs provides a new business model, sensor-as-a-service, for both infrastructure owners and service providers [35].

In previous work on the virtualization of WSNs, virtualization has been studied at two levels: node and network [34]. Node-level virtualization methods enable a single node to process multiple applications concurrently. Approaches to node-level virtualization can be divided into three classes according to the element providing concurrency. These elements are a virtual machine [36], the operating system [37], or middleware [38]. Under network-level virtualization, the virtual network used for running a single application consists of a subset of available sensor nodes. Network-level virtualization results in efficient use of resources because nodes not used by an application can be concurrently used by applications in other virtual networks. Approaches to network-level virtualization can be divided into two classes: overlay-based solutions [13] and cluster-based solutions [11].

Within this system of classification, we aim in this thesis for overlay-based network-level virtualization. Many other researchers have proposed overlay-based approaches to network-level virtualization [13, 33]. Those approaches focus mainly on providing a framework allowing sharing of physical sensor resources. Although improved manageability has been mentioned as a reason for virtualizing networks, methods for constructing a virtual wireless sensor network (VWSN) topology for each application in an overlay layer have not been completely discussed. Moreover, within the IoT environment, changes to traffic patterns, variation in traffic demand, and addition or removal of virtual nodes may all occur. Therefore, providing stable applications remains difficult when using extant approaches in which only required resources are assigned to a user (i.e., approaches without redundancy). Reliability is important even in the face of such environmental change. Accordingly, we focus on a means of constructing a robust and adaptive VWSN topology for applications.

1.2 Outline of this thesis

Potential-based Routing for Supporting Adaptive Any-to-any Communication in WSNs [39, 40]

In Chapter 2, we address an any-to-any routing protocol that offers scalability and adaptivity against environmental changes. For this purpose, we take potential-based routing that has scalability, adaptivity, and energy efficiency [31], and extend it to any-to-any communication. We show that our approach retains the original advantages of potential-based routing.

We first extend potential-based routing to downstream communication for multi-sink WSNs, which retain the advantages of potential-based routing [39]. Our main idea to achieve potential-based downstream routing (PBDR) is to identify a virtual position of a sensor node by trilateration. At least three potential fields thus achieve downstream communication. Then, we achieve a potential-based any-to-any routing protocol (PBAR) by merging potential-based upstream and downstream routing. In PBAR, sensor nodes can send data to a certain node by routing the data via a sink node, which means that the data are first delivered to a sink node by PBUR and then delivered to the destination node by PBDR. We evaluate the data delivery ratio of PBAR at various node

densities and path stretches to show the overhead of our protocol. Here, path stretch is the ratio of the hop count of PBAR to the shortest hop count. We also evaluate our protocol's adaptivity to the failure of multiple sensor nodes and to the failure of a sink node. Using simulation experiments, we show that, given a suitable node density, PBAR attains a data delivery ratio greater than 99.7%. We also show that the data delivery ratio recovers immediately after failure of 30% of sensor nodes and after failure of a sink node. Note that better load balancing is also expected when a potential field that considers residual energy is constructed.

Virtual Wireless Sensor Networks: Adaptive Brain-Inspired Configuration for Internet of Things Applications [41, 42]

In Chapter 3, we consider how to integrate multiple heterogeneous WSNs and how reliable sensor resources could be leased to an application developer. To address this problem, we show an overall architecture that is suitable for constructing and running virtual wireless sensor network (VWSN) services within a VWSN topology. We first propose a method of constructing a VWSN topology that offers robust connectivity against node removal [41]. Our approach is inspired by brain network features, and we focus particularly on the similarity between the hierarchical modular structure of brains and the modular structure that emerges from the integration of local networks. A brain network has many structural properties, including a heavy-tail degree distribution, a high rich-club coefficient, clustering, small-world properties, and a hierarchical modular structure [43]. In particular, small-world properties and a hierarchical modular structure contribute to evolvability of the brain network as well as to high communication efficiency in the global area with low metabolic cost [44]. Our ultimate goal is autonomously evolving VWSNs in response to environmental changes. As a first step toward this goal, we introduce the structural properties of brain networks into a VWSN topology. Our approach consists of two steps: constructing modules with small-world properties, and integrating the modules hierarchically. Our idea is a bottom-up design of the hierarchical structure, inspired by the brain. Modules in a brain network consist of sub-modules that have a close correlation between functions. In our proposed structure, a higher-tier module consists of sub-modules that are assigned to sub-applications such that the module describes an integrated system.

We expect that a modular structure that accounts for modules' functions will contribute to resource-efficient solutions to user demands. In this thesis, however, we evaluate the topology constructed by our proposal without an autonomous configuration because the performance characteristics of the constructed VWSN topology itself are of primary importance. We investigate connection patterns, both within each module and between modules. In our [41], we showed that the VWSN topology constructed by our proposal is robust against node removal selected on both connectivity and path length. Moreover, we show an overall architecture for constructing and running VWSN services on a VWSN topology as a further investigation of our proposed method. Our main focus is providing a user with a reliable VWSN, and we show one solution for providing a user with IoT resources by dividing providers into infrastructural and VWSN providers. Then, we show how to connect modules together to construct a reliable VWSN. We tested this approach by simulation experiment, with the results showing that the VWSN is reliable in many cases, although physical deployment of sensor nodes and the modular structure of the VWSN will be quite important to the stability of services within the VWSN topology. In this, we assume that each application can use a different protocol strategy, and that other protocols can also be configured by users according to their demands. This is clearly consistent with the virtualization scenario.

Percolation Analysis for Constructing a Robust Modular Topology based on a Binary-dynamics Model

From the studies mentioned above, we have found that a connection pattern between modules and the robustness of connectivity of networks is critical for the stability of services. In Chapter 4, therefore, to improve reliability of the VWSN topology proposed in chapter 3, we use an analytical approach and propose a method for characterizing the robustness of a topology according to the method of constructing links between modules. Note that we regard a network topology as an undirected graph in analytical theory. Our approach is to investigate percolation dynamics in a modular network, particularly graph ensembles after addition of inter-module links, to show how to connect modules such that the robustness of connectivity of the constructed network is maximized. Many

researchers have studied percolation processes on various types of graphs by using a generating-function approach [45–48]. In this type of approach, the expected size of the giant component of a random graph ensemble can be found from a probability distribution given as a degree distribution or a distribution of types of links. We can then estimate the robustness of the graph by evaluating percolation transitions of the size of the giant component. Another important analytical method is to use a binary-dynamics model in evaluating percolation and other dynamic processes [49]. This method is relatively simple, and the probability distribution of links can be set independently from the method of percolation. The probability distribution of links represents the modular structure, degree–degree correlation within modules, and degree–degree correlation between modules. The method of percolation can be configured by changing the definition of a response function. The binary-dynamics model can be applied to broad classes of graphs by configuring the probability distribution according to node or link type. These existing studies, however, can be applied only for estimating robustness of given graph ensembles. Therefore, to show robustness of connectivity after addition of inter-module links, we extend a binary-dynamics model [49]. Our proposal enables investigation of percolation analysis according to different embedding patterns of inter-module links when the probability distributions of intra-module graph ensembles and inter-module link ensembles are given independently. Using simulation evaluations, our analytical results are in good agreement with numerical simulations in a configuration model network. Additionally, we show that it is hard to apply our proposal to a graph in which the number of nodes is small. We consider this because the focus of our approach is to compute average properties of random graph ensembles. For a similar reason, we show that the result of analysis cannot completely capture the result of a percolation process on a graph that has special structural properties, such as a ring-shaped structure.

Chapter 2

Potential-based Routing for Supporting Adaptive Any-to-any Communication in Wireless Sensor Networks

2.1 Introduction

Wireless sensor networks have recently attracted attention as a fundamental technology of the Internet of Things, an energy management system, or an emergency system. The cost reduction of sensors has led to and is expected to continue to lead to an increase in the use of wireless sensor networks. In most applications, sensor nodes are supposed to operate in an unattended manner for a long period after their deployment. Owing to the harsh environment of wireless sensor networks and the large number of sensor nodes, it is not uncommon that sensor nodes become faulty and unreliable [14]. Therefore, it is essential that protocols used in wireless sensor networks are adaptive. In this chapter, adaptivity means that the performance recovers even when a node fails. Moreover, energy savings are important in wireless sensor networks because sensor nodes are generally battery-powered.

Further expansion of future applications of wireless sensor networks will diversify the demands of the networks. To meet the requirements of future wireless sensor networks, a sophisticated

any-to-any routing protocol needs to be realized. For an example of such applications, in-network processing such as reactive tasking, data querying, or data-centric storage requires communication between sensor nodes [16]. In addition to typical demands of wireless sensor networks, an any-to-any routing protocol needs to realize low energy consumption, high scalability, robustness, and reliability [17].

During the past few decades, various many-to-one upstream (sensor-to-sink) routing protocols have been studied since an upstream communication pattern and limited resources of sensors are assumed. Many potential-based routing protocols, which are one type of such routing protocols, aim for low overheads, high scalability, and energy balancing [9, 10, 29, 30]. In potential-based upstream routing (PBUR), each node has a scalar value that is called its potential. Each node calculates its potential according to local information, such as the potentials or residual energy of the neighbor nodes. A sensor node whose hop count to a sink is smaller (larger) has a higher (lower) potential. Therefore, if a node sends data to its neighbor node with higher potential, the data will ultimately reach a sink node. Since these potential fields are constructed on the basis of purely local information, PBUR is highly scalable. PBUR is also adaptive because the potential field is changed dynamically and adapts to the condition of the network by updating potentials periodically. Moreover, if these potential fields are constructed according to residual energy, load balancing can be realized.

There is also a demand for downstream(sink-to-sensor) routing [15]. For example, a sink node sends a query to a specific sensor node upon receiving abnormal data from the sensor node, or a sink node sends a message to change the frequency of sensing in a specific domain.

To deliver downstream traffic, we proposed a potential-based downstream routing (PBDR) for multi-sink wireless sensor networks, which retains the advantage of potential-based routing [39]. We showed that PBDR realizes sink-to-sensor routing and meets the requirements of wireless sensor networks because of the adaptive behavior of the potential field.

In this chapter, we realize a potential-based any-to-any routing (PBAR) protocol, which guarantees a high data delivery ratio and adaptive to failure of nodes. Any-to-any routing is accomplished by merging PBUR and PBDR. The data flow of PBAR is shown in Figure 2.1. When each sensor node needs to send data to a certain node, the data are first delivered to a sink node by PBUR and

then delivered to their destination node by PBDR.

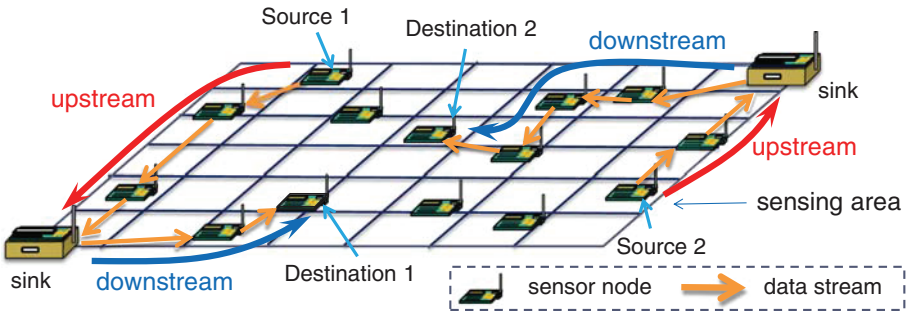


Figure 2.1: Data flow of PBAR via a sink node.

We evaluate the data delivery ratio of PBAR at various node densities and path stretch, which is the ratio of the hop count of PBAR to the shortest hop count, to show the overhead of our protocol. We also evaluate our protocol's adaptivity to the failure of multiple sensor nodes or the failure of a sink node. Note that communication between arbitrary nodes can be realized without going through a sink node once both nodes retain each other's virtual coordinates.

Our proposal can be applied to various networks, such as an ad-hoc network. However, our proposal has the drawback that it cannot guarantee a data delivery ratio of 100% because our main focus is to realize any-to-any communication in WSNs where a large number of low-spec nodes exist. Therefore, our proposal is a candidate for a routing protocol in a network where a large number of low-spec nodes exist.

The rest of the chapter is organized as follows: we start by giving an overview of related work in Section 2.2. Sections 2.3 and 2.4 respectively show the potential-based upstream and downstream routing protocols. We present the proposed PBAR protocol in Section 2.5. In Section 2.6, we evaluate the performance of PBAR through simulation experiments. Finally, Section 2.7 gives our summary.

2.2 Related Work

For wireless ad hoc sensor networks, various any-to-any routing protocols have been studied. In the flooding method and gossiping method, messages are relayed on the basis of broadcasts [18, 19].

These methods suffer from a high number of redundant transmissions, particularly when a few nodes in a specific domain are the destinations.

Many studies have been conducted on reactive and proactive routing protocols [20, 21]. In reactive protocols, each node constructs routes only in the case where communication is required. Power consumption can then be cut when communication is not needed. The delay time, however, is longer for reactive protocols because of their route discovery procedures. This means that reactive protocols are not appropriate for real-time applications. In proactive protocols, end-to-end delay is small. However, there is overhead because all the nodes collect information about links.

Geographic routing protocols allow for communication between two arbitrary nodes [22]. Equipment for acquiring the precise geographic position is required for these protocols, and all the nodes must know the position of their destinations. The virtual coordinate assignment protocol (VCap) is able to route data using a virtual position without the need for Global Positioning System devices [23]. In VCap, all nodes have three shortest hop counts from three anchor nodes and use them as virtual coordinates. Note that since the hop count is an integer, some nodes may have the same virtual coordinates in VCap. Many other virtual coordinate assignment protocols have been proposed [24–28].

PBUR protocols are categorized as being proactive. In PBUR, all the nodes have a scalar potential that constructs a potential field. Each node updates its potential according to local information, such as the potentials of its neighbors, its residual energy and that of its neighbors, or the hop count to a sink node. A sensor node whose hop count to a sink is smaller (larger) has a higher (lower) potential. Each node with data to be sent forwards the data to a node whose potential is higher than its own, and the data ultimately reach the sink node. Moreover, load balancing and extending the lifetime of wireless sensor networks using the residual energy of neighbor nodes or the amount of traffic have been studied [9, 10, 31].

In this chapter, we propose any-to-any routing based on PBUR and PBDR. PBAR, like PBUR, is a type of proactive routing, and PBAR has scalability because it is realized through local information exchange only. Additionally, our method achieves a high data delivery ratio and adaptivity to failure of nodes. Better load balancing is also expected when a potential field considering a residual energy is constructed.

In the simulation experiment carried out in this chapter, a potential field constructed using our method is based on a controlled potential-based routing (CPBR [31]). However, our method is applicable to any strategy for potential field construction. We give details of PBUR in the following section.

2.3 Potential-based upstream routing protocols

Potential-based routing delivers data along the gradient of the potential field constructed over a wireless sensor network. A potential is a scalar value (like electric potential) and calculated by each sensor node from local interactions using the potential of its neighbors, its residual energy and that of its neighbors, or the hop count to a sink node. Smaller (larger) hop count from a sink node leads a higher (lower) potential to a sensor node. Thus, the gradient of a potential field means the direction to a sink node.

CPBR [31] constructs a potential field for multisink wireless sensor networks using the diffusion equation (Equation 2.1). The equation provides the magnitude ϕ of the diffusing quantity at time t and position X:

$$\frac{\partial \phi(X,t)}{\partial t} = D\Delta\phi(X,t), \tag{2.1}$$

where D is the diffusion rate and takes a positive value. By discretizing this equation and regarding ϕ as a potential, it is possible to construct a potential field from local information only.

A discrete form of the diffusion equation is described as Equation2.2. $\phi(n,t)$ describes the potential of node n at time t. $\mathcal{N}(n)$ is a set of nodes neighboring node n. A parameter $D(n)$ changes the magnitudes of influences of the potentials of the neighbor nodes. It is noteworthy that potentials may oscillate when $D(n)$ is large. In CPBR, $D(n)$ is set to $\frac{\epsilon}{|\mathcal{N}(n)|}$ to keep the potential from oscillating, where $|\mathcal{N}(n)|$ is the cardinality of the set $\mathcal{N}(n)$. It is then considered that each node has been affected by the potential of essentially only one node. As a result, ϵ is set to a value between 0 and 1 to keep the potential from oscillating.

$$\phi(n,t+1) = \phi(n,t) + D(n) \sum_{\nu \in \mathcal{N}(n)} \left(\phi(\nu,t) - \phi(n,t)\right). \tag{2.2}$$

Figure 2.2 shows the shape of the potential field after convergence. In CPBR, the potential of each sink node is calculated according to the number of data received by each sink node to realize load balancing. The potential of each sensor node deployed at the boundary of the network is set to the minimum potential so that the potentials of all nodes will not eventually arrive at a value much the same as the potential of the sink node. The potential of each of the other sensor nodes converges to the average value of the potentials of the neighbor nodes by using Equation 2.2. The potential field is then constructed, and monotonicity from a sensor node deployed at the boundary of the network to a sink node is ensured.

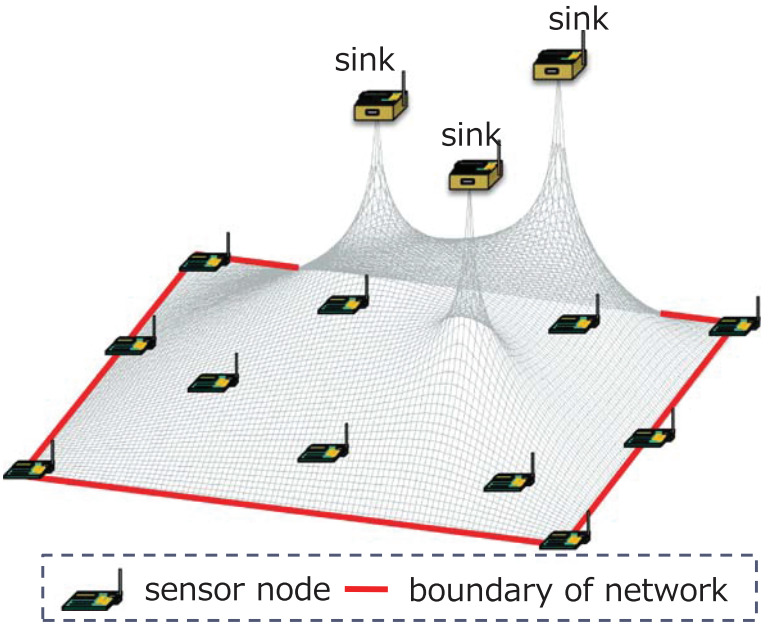


Figure 2.2: The shape of a potential field constructed by CPBR.

In existing PBUR protocols, there is a possibility that some sensor nodes have the same potential because PBUR guarantees uniqueness of only a potential assigned to a sink node. Therefore, when the sink node transmits data to a certain sensor node along the gradient of the potential field constructed through existing PBUR protocols, the data will not always arrive at the destination. This problem is treated as a contour problem, as shown in Figure 2.3. The contour problem means that no node can determine a next hop because no node knows the geographic direction to its destination node from the potentials.

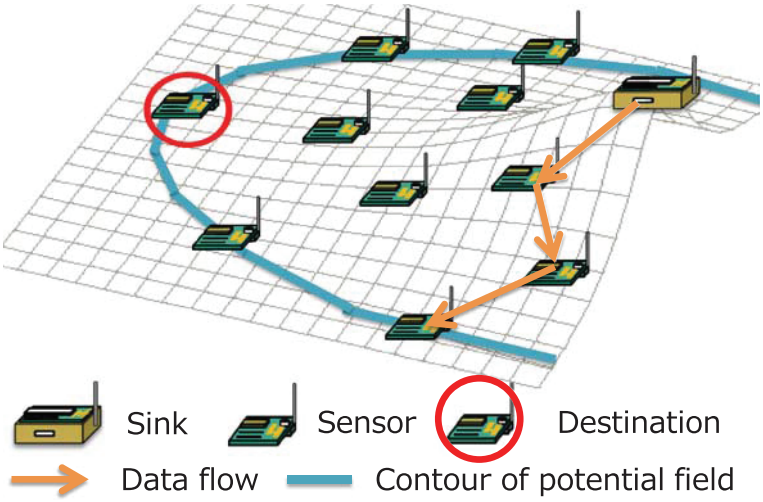


Figure 2.3: Contour problem for downstream routing using an existing potential construction method.

We focused on the advantages of PBUR for wireless sensor networks and implement downstream routing by extending PBUR. The details of PBDR are given in the following section.

2.4 Potential-based downstream routing

PBDR, which we proposed in [39], must accomplish the following three tasks to handle the contour problem:

1. Assign potentials to all sensor nodes to identify them
2. Inform the sink nodes of the potentials
3. Route data to a destination node using its potential as a virtual location

In the following PBDR algorithm, we suppose that all sinks can communicate with each other via the wired link. In Section 2.4.1, we present the overview of our method. The node identification algorithm is then presented in Section 2.4.2. We show how to manage virtual coordinates of destination nodes in Section 2.4.3 and the downstream routing algorithm in Section 2.4.4. In our protocol, the local-minimum problem arises when a node cannot select next hop. In Section 2.4.5,

we explain this problem and how to solve it. Finally, the media access control (MAC) layer protocol we use is presented in Section 2.4.6.

2.4.1 Overview of PBDR

To realize PBDR, it is first necessary to assign potentials to all sensor nodes to identify them. We denote such a potential as P_{id}, and we give an overview of PBDR with P_{id} below:

1. Node identification: Each sensor node calculates its own P_{id}.

2. Management of P_{id}: When a sensor node generates an upstream data packet, it includes its P_{id} in the packet header. A sink node sends the upstream data to a system on the user's terminal when it receives the upstream data. The system on the user's terminal then records the P_{id}.

3. Downstream routing: We define a function $Dist_p(n_1, n_2)$ that is a virtual distance between nodes n_1 and n_2 and is calculated from their P_{id}s. A sensor node with downstream data to be transmitted forwards the data to the neighbor node whose distance to the destination node is smallest, as shown by the value of function $Dist_p(n_1, n_2)$. In this way, the data ultimately reach the destination node.

4. Local-minimum problem: A node cannot select a next hop node for downstream data when the local-minimum problem arises. We resolve the local-minimum problem by using another virtual distance function when the local-minimum problem occurs.

2.4.2 Node identification

In protocols based on existing methods for constructing a potential field, downstream data will not always arrive at the destination node because of the contour problem. Thus, we assign a virtual coordinate to all sensor nodes to identify them. This method is based on the idea of trilateration. α sink nodes individually construct potential fields, and all nodes have a set of potentials as a virtual coordinate. Here, as in [31], the diffusion equation is used by sink node i to construct the potential field $PF_i(i = 1, \cdots, \alpha)$. We can now define that P_{id} is a set of α potentials and use P_{id} as a

destination address. If there are at least three sink nodes and three potential fields, PBDR can be realized. In Section 2.6, we use four sink nodes and four potential fields to acquire redundancy in case a sink node fails.

Equation 2.3 is used to construct the potential field PF_i having potential $\phi(n,t,i)$ at node n and time t. ϵ is a constant that plays the same role as ϵ in Equation 2.2.

$$\phi(n,t+1,i) = \phi(n,t,i) + \frac{\epsilon}{|\mathcal{N}(n)|} \sum_{\nu \in \mathcal{N}(n)} \left(\phi(\nu,t,i) - \phi(n,t,i)\right). \tag{2.3}$$

Generally, in the diffusion equation, when all boundary conditions have the same value, all values in the field converge to the value of the boundary conditions, and the field eventually becomes flat. Consequently, potential routing does not work because there is no gradient in the field without a boundary condition. Therefore, we use Equation 2.4 as a boundary condition so that the potentials of the entire network do not converge to the potential of a sink node. S is a set of sink nodes. Note that sink node i constructs the potential field PF_i.

$$\forall s \in S, \phi(s,t,i) = \begin{cases} \phi_{\max} & \text{if } i = s \\ \phi_{\min} & \text{otherwise.} \end{cases} \tag{2.4}$$

2.4.3 Management of P_{id}

As mentioned in Section 2.4.2, we use multiple sink nodes and multiple potential fields to acquire redundancy in case a sink node fails. We assume that all sink nodes are connected to the user's terminal via the wired link and can communicate with each other. When a sink node receives upstream data, it sends the data to the user's terminal, and a system on the user's terminal records the tuple {source node's ID, source node's P_{id}, generation time of the data} in its look-up table. When a user wants to send a query or a control message, the user commands the system to send downstream data and the system selects the sink node that is closest to the destination. The selected sink node then starts to send the downstream data.

We show the upstream and downstream packet formats in Tables 2.1 and 2.2, respectively. In upstream routing, a source node includes its P_{id} in a packet header, and sinks obtain the P_{id} when

they receive the packet. In downstream routing, a sink includes P_{id} of the destination in a packet header, and relay nodes can refer to the P_{id} of the destination. The time to live (TTL) is the maximum number of hops that data can be forwarded. A loop flag for a downstream packet is used in checking whether the data is in a loop, and we explain how to use this in Section 2.4.5.

Table 2.1: Upstream packet format

Source node ID (2byte)	Destination node ID (2byte)
Sender node ID (2byte)	Receiver node ID (2byte)
Sequence number (2byte)	TTL (1byte)
P_{id} (source) (16byte)	
Sensing data (72byte)	

Table 2.2: Downstream packet format

Source node ID (2byte)	Destination node ID (2byte)	
Sender node ID (2byte)	Receiver node ID (2byte)	
Sequence number (2byte)	TTL (1byte)	loop flag (1byte)
P_{id} (destination) (16byte)		
Query or control message (72byte)		

2.4.4 Downstream routing

As we have described in the previous section, we assume that a user's terminal and each sink node can communicate with each other via the wired link, and a user commands the system to send downstream data. A downstream data packet can then be routed to the sink node closest to a destination node, and the sink node can start delivery of the downstream data.

We define potential distance as the virtual distance calculated from P_{id}. To select a next hop, node n calculates the potential distance $Dist_p$ between its neighbor $\nu(\in \mathcal{N}(n))$ and destination node d:

$$Dist_p(\nu, d) = \sqrt{\sum_{i=1}^{\alpha} (PF_i(\nu) - PF_i(d))^2}, \tag{2.5}$$

where $PF_i(\nu)$ is the potential of node ν in the potential field PF_i, and $PF_i(d)$ is the potential of destination node d. We use potential distance as a routing metric. A sink node includes P_{id} of destination node d in the header of a downstream data packet, and relay nodes forwards the data to node n_1 that fulfills the following condition:

$$n_1 = \arg\min_{\nu \in \mathcal{N}(n)} Dist_p(\nu, d). \tag{2.6}$$

For the sake of greater reliability, our downstream routing allows a sender node to forward data to the neighbor that is the second closest to the destination node. Two neighbor nodes n_1 and n_2 are then candidates for the next hop node. In Section 2.4.6, we explain how to select the next hop node from n_1 and n_2 in detail. When data reach a neighbor node of the destination node, they are forwarded to the destination node using a node ID.

2.4.5 Local-minimum problem

In the local-minimum problem, a sender node has no neighbor node whose $Dist_p$ is smaller than that of the sender. This problem occurs around void areas, as is well known in geographic routing [50]. Once the problem arises, sender nodes cannot forward data.

In the example shown in Figure 2.4, node C must forward data to node D so that the destination node receives the data. However, $Dist_p(D, destination)$ is larger than $Dist_p(B, destination)$, and node C does not forward data to node D. We use a local detour rule, by which node v forwards data to node w having the smallest $Dist_p(w, destination)$, even if $Dist_p(v, destination)$ is smaller than $Dist_p(w, destination)$, and node v does not forward data to node u after node v receives the data from node u. According to this rule, node C forwards the data to node B. As a result, a data packet will follow a loop through node A, node B, and node C.

The local-minimum problem occurs when a destination node is near the boundary of the monitoring area. This is because the node density near the boundary of the monitoring area is low, which leads to a void area. Hence, we assume that a destination node exists near the boundary of the monitoring area when a loop is detected. We then resolve the local-minimum problem using an alternative routing metric.

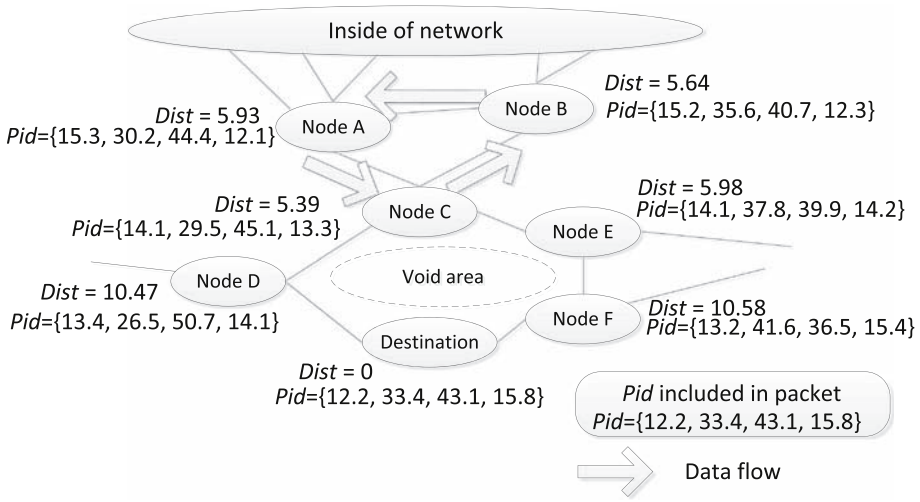


Figure 2.4: Local-minimum problem.

The main idea to solve this problem is using only one potential field when a loop is detected. Because the diffusion equation is a harmonic function, a loop rarely occurs when a single potential field is used for downstream routing. The node near the monitoring area boundary is located in the area farthest from a certain sink node, and the potential of the destination node in the potential field built by the sink node is nearly equivalent to $\phi_{\min}$ by using Equation 2.3. Thus, the possibility that the data packet gets close to the boundary of the monitoring area is high when a node forwards the packet to the node farthest from the sink node. To send data to the boundary of the monitoring area, we use the potential field whose potential is the smallest in P_{id} of the destination node. From the above, we define a potential gap $Gap(\nu, d)$ (Equation 2.7) and use it as an alternative routing metric when a routing loop is detected. $Gap(\nu, d)$ is a potential distance between node ν and destination d on the potential field, PF_i, where the potential of the destination is the smallest among all $PF_j (1 \leq j \leq \alpha)$. Then, a node which detects a loop calculates potential gaps of its neighbors and forwards data to the neighbor whose potential gap is the smallest among them.

$$Gap(\nu, d) = |PF_i(\nu) - PF_i(d)|, i = \arg\min_{1 \leq j \leq \alpha} PF_j(d). \tag{2.7}$$

For example, in Figure 2.4, the potential gap of node A is 3.1, that of node C is 1.9, and that of

node D is 1.2. Node C then forwards the data to node D, and the data reach the destination node.

A sequence number and a loop flag are included in the data packet header and are used to detect routing loops. When a node receives a downstream data packet, the node records the sequence number of the data. When a node receives data with the same sequence number, the node judges that a loop has occurred and sets the loop flag to one. Each node records n_{hist} sequence numbers of received packets from the newest received one.

In addition to the case where $Dist_p$ is used, when Gap is used, our downstream routing allows a sender node to forward data to the neighbor that is the second closest to the destination node. In Algorithm 1, we show how to determine two candidates for a next hop node. How to select a next hop node from those two nodes is shown in detail in the following section.

2.4.6 MAC layer protocol

In this chapter, we use intermittent receiver-driven transmission (IRDT) for the MAC layer protocol [51]. In IRDT, all nodes sleep and wake up asynchronously with the duty cycle T_{duty}. Whenever a node wakes up, it sends an ID message that informs neighbor nodes that the node is ready to receive data.

When node n_s forwards data to node n_r in IRDT, the procedure shown in Figure 2.5 is used. Node n_s with data to be sent wakes up and waits for an ID message from node n_r. Upon receiving an ID message from node n_r, node n_s sends an SREQ message, which is a communication request informing node n_r that it has a data packet for node n_r. When node n_r receives the SREQ message, it stays awake and sends to node n_s a RACK message, which is an acknowledgement of the communication request. Afterward, node n_s sends a data message to node n_r. Finally, node n_r sends to node n_s a DACK message, which is an acknowledgement of the data. If node n_r is not a destination node, node n_r becomes a sender and waits for an ID message from a neighbor node.

Node n_s drops the data when forwarding the data does not succeed within $T\mathrm{timeout}$ after node n_s wakes up. Additionally, when the number of forwardings exceeds the TTL, node n_s drops the data.

In our protocol, there are two candidates for a next hop, n_1 and n_2, for obtaining reliability. n_1

Algorithm 1 Select a next hop for downstream routing

Require: Node n receives a data packet whose destination node is node d

Ensure: Return two candidates for a next hop node that have the smallest or second smallest $Dist_p$ or Gap to the destination

if loop_flag of the received data packet equals to one **then**
  $next\_hop.address1 \Leftarrow getNextHopUsingGap(d, n)$
  $next\_hop.gap1 \Leftarrow getGap(d, next\_hop.address1)$
  $next\_hop.address2 \Leftarrow getSecondNextHopUsingGap(d, n)$
  $next\_hop.gap2 \Leftarrow getSecondGap(d, next\_hop.address2)$
  if $Gap(n, d) < next\_hop.gap1$ **then**
    loop_flag of the received data packet is set to zero
    $next\_hop.address1 \Leftarrow getNextHopUsingDist_p(d, n)$
    $next\_hop.dist1 \Leftarrow getDist_p(d, next\_hop.address1)$
    $next\_hop.address2 \Leftarrow getSecondNextHopUsingDist_p(d, n)$
    $next\_hop.dist2 \Leftarrow getSecondDist_p(d, next\_hop.address2)$
  end if
else
  if sequence number of the received data packet is in the set of node n's history **then**
    loop_flag of the received data packet is set to one
    $next\_hop.address1 \Leftarrow getNextHopUsingGap(d, n)$
    $next\_hop.gap1 \Leftarrow getGap(d, next\_hop.address1)$
    $next\_hop.address2 \Leftarrow getSecondNextHopUsingGap(d, n)$
    $next\_hop.gap2 \Leftarrow getSecondGap(d, next\_hop.address2)$
  else
    $next\_hop.address1 \Leftarrow getNextHopUsingDist_p(d, n)$
    $next\_hop.dist1 \Leftarrow getDist_p(d, next\_hop.address1)$
    $next\_hop.address2 \Leftarrow getSecondNextHopUsingDist_p(d, n)$
    $next\_hop.dist2 \Leftarrow getSecondDist_p(d, next\_hop.address2)$
  end if
end if
return $next\_hop$.

is the first closest node and n_2 is the second closest node to the destination. Here, we explain how to select a next hop node from n_1 and n_2. When a sender node n_s receives an ID message from n_1 or n_2, n_s stochastically determines whether or not it returns an SREQ message to the sender of the ID regarding it as a next hop node. When $Dist_p$ is used for a potential distance metric, the probability of selecting n_1 is 1, and that of selecting n_2 is $\frac{Dist_p(n_s,d)+Dist_p(n_1,d)}{Dist_p(n_s,d)+Dist_p(n_2,d)}$.

The probability of selecting n_2 is close to 1 when the difference between $Dist_p(n_1, d)$ and $Dist_p(n_2, d)$ is small. In such a case, both nodes n_1 and n_2 are suitable to be the next hop because the distance to the destination node is almost the same. We add $Dist_p(n_s, d)$ to the numerator and denominator so as to provide multipath even when $Dist_p(n_1, d)$ is almost zero. Similarly, when Gap is used, the probability of selecting n_1 is 1 and that of selecting n_2 is $\frac{Gap(n_s,d)+Gap(n_1,d)}{Gap(n_s,d)+Gap(n_2,d)}$.

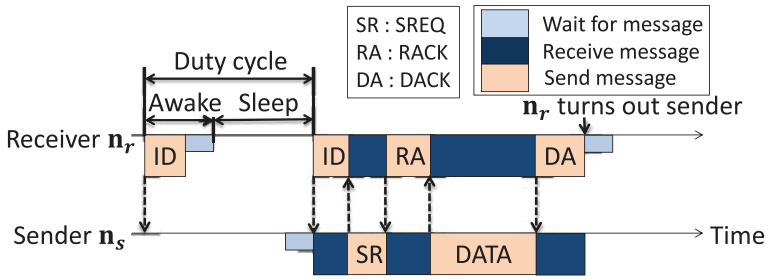


Figure 2.5: Procedure of forwarding data in IRDT.

2.5 Potential-based any-to-any routing

2.5.1 Outline

As mentioned in Section 2.1, we realize PBAR by merging PBUR and PBDR. We assume that each sensor node only has its own and its neighbors' P_{id}s, and each sink node has all sensor nodes' P_{id}s. This means that each sensor node does not have the destination node's P_{id} when generating data for a certain node. Therefore, in PBAR, the data are delivered to a sink node first, and the sink node then includes the destination's P_{id} in the header of the data. The following is an outline of PBAR:

1. A source node sends data to a sink node through PBUR when it generates data for a certain sensor node.

2. When a sink node receives the data, it sends the data to the sink node that is closest to the destination node in terms of potential distance via the wired link. The sink node then starts to send the data to the destination node through PBDR.

2.5.2 Any-to-any routing

First, each sink node constructs its own potential field, and all nodes have a set of potentials as a virtual coordinate. After convergence of P_{id}s, each sensor node generates upstream data packets containing its P_{id} and forwards them to a sink node through PBUR. When a sink node receives the data, the node broadcasts the data to the other sink nodes via the wired link, and all sink nodes can record the P_{id}s of all sensor nodes. Afterward, when a sensor node generates data destined for a certain sensor node, the data are delivered to the closest sink node through PBUR.

Note that not all data have to go through a sink node. When the destination node is a neighbor node of one of the relay nodes, the relay node can send the data to the destination node directly.

In the method described above, much data has to go through one of the sinks, and this may result in large path stretch. However, in some cases, the number of hops until the data arrival at the destination node may be less than that of the shortest hop path. Figure 2.6 shows one example of such cases. In this example, the wired link between sink nodes can be used as a shortcut link. When data generated by the source node are forwarded along the shortest hop path, they go through nodes A, B, and C before arriving at the destination node. Here, the number of hops is four. In PBAR, the source node first forwards data to sink A. Sink A then sends the data to sink B via the wired link because sink B is the closest sink to the destination node. Finally, sink B forwards the data to the destination. Therefore, the number of hops is two for wireless communication and one for wired communication. Because there are fewer wireless communications between sensor nodes, sensor nodes can save their energy.

2.5.3 Constraints

In our proposed method, at least three sink nodes must be deployed in the sensing area. To identify each sensor node, a virtual coordinate needs to be assigned to a sensor node uniquely. When

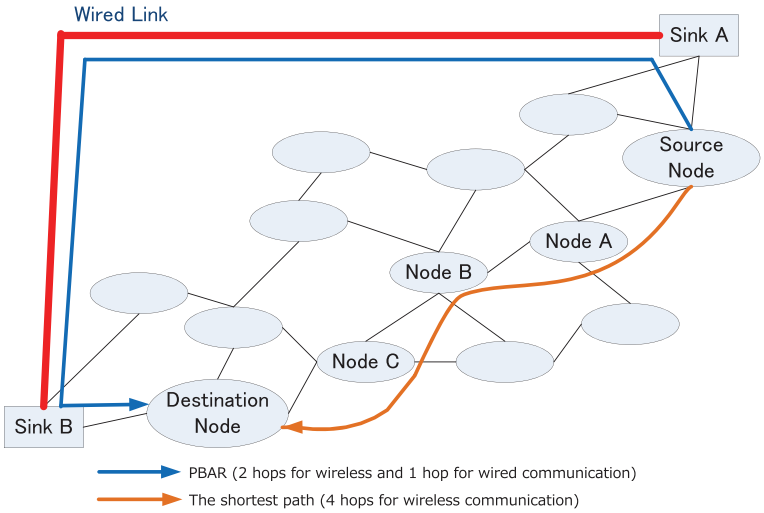


Figure 2.6: An example showing smaller hop number in PBAR than in the shortest hop routing.

less than three sink nodes are deployed and each of them constructs its own potential field, some sensor nodes may have the same virtual coordinate because our method is based on a theory of triangulation. In such a case, data may be routed to a sensor node that is not a destination node.

To send data to a specific sensor node, sink nodes must know the P_{id} of a destination node. Therefore, each sensor node needs to send data to a sink node periodically. In this chapter, each sensor node uses the potential field with the highest potential at its P_{id} to send upstream data to the nearest sink node. A source node inserts its P_{id} into the header of the packet, and sink nodes can collect P_{id} for each sensor node. Each sink node discards a P_{id} when it does not receive a new one from a sensor node for $T_{\text{expiration}}$. A data packet is dropped when a sink node tries to send the packet to a destination node but no sink node has P_{id} of the destination node. This means that the sink nodes temporarily could not receive any upstream data from the destination node. Such a packet drop occurs when an upstream data packet is dropped, when the delay is large, or when a destination node is isolated.

To realize PBAR, deployed sink nodes need to be connected to each other with a high-speed link. Otherwise, sink nodes could not share all P_{id}s and downstream data could not be routed to the sink node that is closest to the destination node.

In our proposed method, many data may be dropped because of congestion. When node A has

data to be forwarded to node B and node B has data to be forwarded to node A at the same time, neither node can forward the data. This leads to small throughput or to the dropping of data due to timeout. Hence, many data may be dropped when traffic is heavy and congested. This influence becomes remarkable around the sink nodes because all data go through a sink node. When the queue of a sink node is full with downstream data, a neighbor node of the sink node cannot forward upstream data to the sink node, and vice versa. Thus, many nodes around the sink node cannot forward data unless some data are dropped. Therefore, our proposed method targets the situation of comparatively low traffic load. However, this constraint can be loosened easily by enlarging the queue size of nodes.

2.6 Simulation experiments

In this section, we present the results of our simulation experiments. PBAR is implemented on the OMNeT++ [52] network simulator. We evaluate our method in two situations: one where all the data go through a sink node and the other where a relay node of which the destination node is a neighbor sends the data to the destination node directly. We denote the former as situation 1 (S1) and the latter as situation 2 (S2). We evaluate the data delivery ratio and the path stretch of PBAR at various node densities in S1 and S2.

The sensor nodes are randomly distributed in a 600m $\times$ 600 m square. In this network, the number of deployed sensor nodes is from 50 to 250, and four sink nodes are situated at the four corners of the observation area. The rate of data generation is $\frac{1}{300}$ per sensor node for any-to-any communication in a Poisson process. The model of radio attenuation is the free-space model [53], and we assumed that no noise exists. Each sensor node selects a destination node randomly and starts to send the data to the destination node through PBAR when it generates the data. The queue size of each node is one, and a node with data does not broadcast an ID message. The other parameter settings are summarized in Table 2.3. Under these conditions, we evaluate how the data delivery ratio is affected by node density (Section 2.6.1), by sensor node failure (Section 2.6.2), and by sink node failure (Section 2.6.3).

Table 2.3: Simulation configuration

parameter	Radio range	TTL	Data size	other message size	Bandwidth
value	100m	30	100 byte	28 byte	100kbps
	$\phi_{\max}$	$\phi_{\min}$	ϵ	n_{hist}	
	90	0	0.8	3	
	T_{duty}	T_{update}	T_{timeout}	$T_{\text{expiration}}$	
	1s	100s	5s	2,500s	

2.6.1 Data delivery ratio

Simulation results for S1 are shown in Tables 2.4 and 2.5. The number of trials is 50, and the confidence interval is 95%. The simulation time is 30,000 s. In those tables, Drop_{TTL}, $\text{Drop}_{\text{timeout}}$, and $\text{Drop}_{\text{noinfo}}$ mean the packet drop ratio when the number of forwarding of data exceeds the TTL, when a node with a data packet cannot forward the data within T_{timeout} after the node generates or receives it, and when no sink node has P_{id} of the destination node, respectively.

Table 2.4: Data delivery/drop ratio for S1

Number of sensor nodes	Node density	Data delivery ratio (%)	Drop_{TTL} (%)	$\text{Drop}_{\text{timeout}}$ (%)	$\text{Drop}_{\text{noinfo}}$ (%)
50	Lower	99.33 ± 0.19	0.584	0.053	0.023
100	Lower	99.67 ± 0.07	0.203	0.098	0.021
150	Medium	99.77 ± 0.02	0.016	0.183	0.025
200	Higher	99.07 ± 0.09	0.007	0.894	0.023
250	Higher	97.65 ± 0.09	0.007	2.312	0.024

The data delivery ratio is low when the node density is low because there are few links in the entire network and the local-minimum problem thus easily occurs. This is clear from the fact that Drop_{TTL} is comparatively high when the number of sensor nodes is 50 or 100. The data delivery ratio is high when the node density is high because there are more links in the entire network. When the node density is excessively high, however, packet collisions and congestion occur frequently, especially near sink nodes, thus decreasing the data delivery ratio. This is shown by the fact that $\text{Drop}_{\text{timeout}}$ is comparatively high when the number of sensor nodes is 200 or 250. The data

Table 2.5: Path stretch for S1

Number of sensor nodes	Path stretch
50	2.88
100	3.53
150	3.40
200	3.78
250	4.01

delivery ratio is highest when the number of nodes is 150. In that case, the data delivery ratio is 99.7% and the average number of neighbor nodes is 16.7. As shown in Table 2.5, the average number of hops is approximately three to four times that of the shortest hop path. This is because all data go through a sink node. However, the path stretch can be decreased when a relay node that is a neighbor of the destination node sends the data to the destination node directly.

Simulation results for S2 are shown in Tables 2.6 and 2.7. As shown in Table 2.6, the data delivery ratio and the drop ratio have the same characteristic as those shown in Table 2.4. The data delivery ratio is low when the node density is low or excessively high. However, in comparison with Table 2.4, $\mathrm{Drop}_{\mathrm{timeout}}$ is lower and $\mathrm{Drop}_{\mathrm{noinfo}}$ is higher. The reason for the former is that the traffic load around sink nodes is low because not all the data have to go through a sink node, and the reason for the latter is that sink nodes may not receive the data and update the sensor nodes' P_{id}s. When a sink node cannot update the sensor nodes' P_{id}s for a long time, it discards the P_{id}s.

Table 2.6: Data delivery/drop ratio for S2

Number of sensor nodes	Node density	Data delivery ratio (%)	$\mathrm{Drop}_{\mathrm{TTL}}$ (%)	$\mathrm{Drop}_{\mathrm{timeout}}$ (%)	$\mathrm{Drop}_{\mathrm{noinfo}}$ (%)
50	Lower	99.34 ± 0.16	0.485	0.058	0.116
100	Lower	99.58 ± 0.07	0.175	0.073	0.165
150	Medium	99.70 ± 0.02	0.017	0.161	0.109
200	Higher	99.20 ± 0.05	0.005	0.656	0.128
250	Higher	98.19 ± 0.07	0.005	1.665	0.135

Comparing Table 2.7 with Table 2.5, the path stretch is smaller in S2. The average number

Table 2.7: Path stretch for S2

Number of sensor nodes	Path stretch
50	2.32
100	2.74
150	2.78
200	2.99
250	3.17

of hop is as much as three times that of the shortest hop path. Note that the path stretch can be about one once the source node retains the destination node's virtual coordinate. This is because relay nodes can calculate the potential distance and find the closest next hop to the destination node without data going through a sink node when the source node includes the destination node's virtual coordinate in the header of the data.

Because of an imperfect routing table, PBAR cannot guarantee the data delivery ratio to be 100%. However, an improvement is possible. A larger queue size that allows a node to forward data quickly may reduce $\mathrm{Drop}_{\mathrm{timeout}}$. $\mathrm{Drop}_{\mathrm{noinfo}}$ can be alleviated when each node sends upstream data that contains its P_{id} to a sink node periodically. Although $\mathrm{Drop}_{\mathrm{TTL}}$ can hardly be reduced owing to the local-minimum problem, other next hop decision strategies or the retransmission of the upperlayer protocol allows a high data delivery ratio.

2.6.2 Failure of sensor nodes

In the case where 150 nodes and 45 sensor nodes fail, we evaluate the data delivery ratio from $t - 1,000$ (s) to t (s) at each time t. The simulation time is 80,000 s, and 45 sensor nodes fail after 40,000 s have elapsed. The number of trials is 10. We assume that the situation is S1.

Figure 2.7 shows the data delivery ratio and the drop ratio from $t-1,000$ (s) to t (s) at each time t. The results show that our proposed routing works well even if sensor nodes fail. The data delivery ratio decreases steeply when sensors fail at 40,000 s but quickly returns to the level observed before the failure of nodes. $\mathrm{Drop}_{\mathrm{TTL}}$ and $\mathrm{Drop}_{\mathrm{noinfo}}$ do not change considerably when sensor nodes fail, but $\mathrm{Drop}_{\mathrm{timeout}}$ increases steeply. This is because a sensor node cannot select the next hop that is

closer to the destination node until the potential fields converge.

From this result, PBAR is adaptive to the failure of sensor nodes. Even after 30% of sensor nodes fail, the data delivery ratio is more than 93.1%. It takes about 1,200 s for the data delivery ratio to recover to 99% after the sensor nodes fail, which relates to the period in which each node updates its potential. Therefore, when each sensor node updates its potential more frequently, it takes less time to recover the data delivery ratio, while the energy consumption increases.

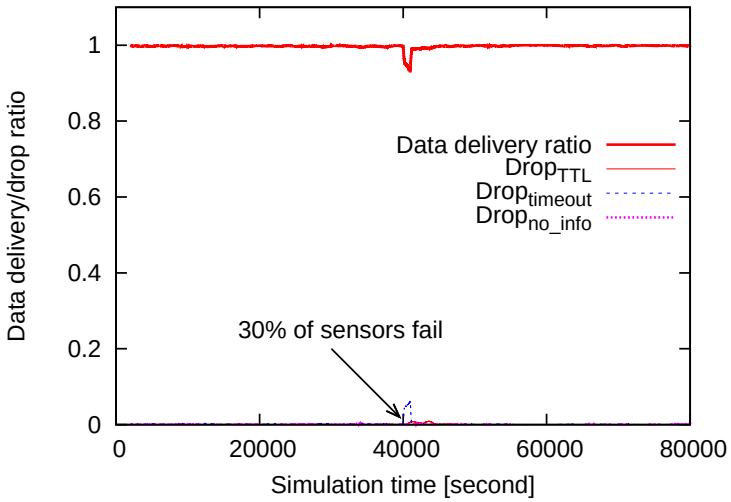
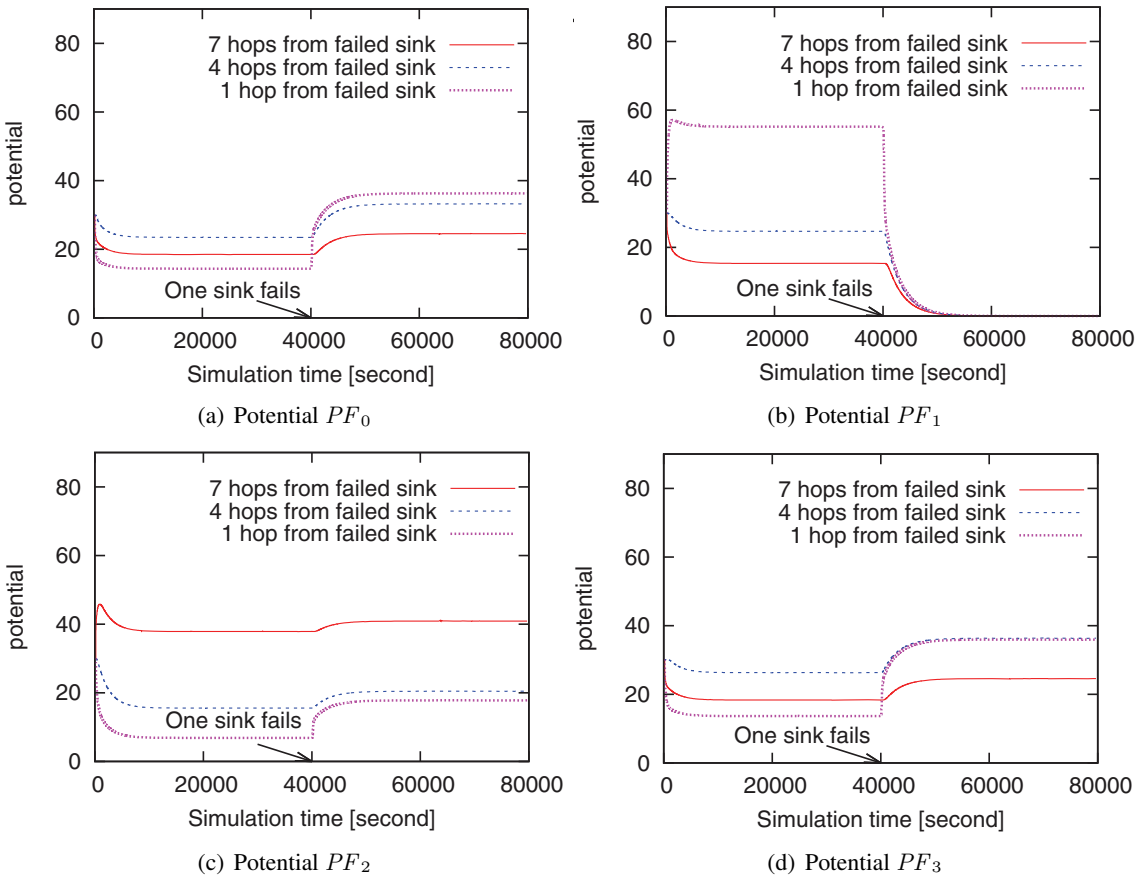


Figure 2.7: Data delivery/drop ratio in the case where sensor nodes fail.

2.6.3 Failure of a sink node

In the case where 150 nodes and one of four sink nodes fails, we evaluate the data delivery ratio immediately prior to 1,000 s. The simulation model is the same as that for sensor node failures. When sink node s fails, all the potentials in PF_s converge on $\phi_{\min}$ because of the boundary condition (2.4). A sensor node with upstream data to be sent decides the next hop according to the potential field whose value is highest among the potentials. In this manner, the other three sink nodes collect the P_{id} for each sensor node and PBAR regains its effectiveness after the sink node failure.

Figure 2.8 shows changes in potential until the potential fields converge. Here, the changes in potential for three nodes are shown. The first node is the farthest from the failed sink node, with a hop count of seven to the failed sink node. The second is deployed near the center of the network, with a hop count of four to the failed sink node. The third is a one-hop neighbor of the failed sink node. In Figure 2.8(b), the changes in the potential field constructed by the failed sink node are shown and the potentials converge to $\phi_{\min}(=0)$ in about 30,000 s. In Figure 2.8(a), 2.8(c),2.8(d), the changes in the potential fields that the other three sink nodes construct are shown and the potentials converge in about 20,000 s.



(a) Potential PF_0

(b) Potential PF_1

(c) Potential PF_2

(d) Potential PF_3

Figure 2.8: Potential convergence after sink node failure.

The data delivery ratio and the drop ratio from $t - 1,000$ (s) to t (s) at each time t are shown in Figure 2.9. In Figure 2.9, the data delivery ratio decreases steeply when one of the sink nodes

fails at 40,000 s but quickly recovers to the level observed before the failure. $\mathrm{Drop}_{\mathrm{noinfo}}$ does not change considerably when a sink node fails, but $\mathrm{Drop}_{\mathrm{TTL}}$ and $\mathrm{Drop}_{\mathrm{timeout}}$ increase steeply.

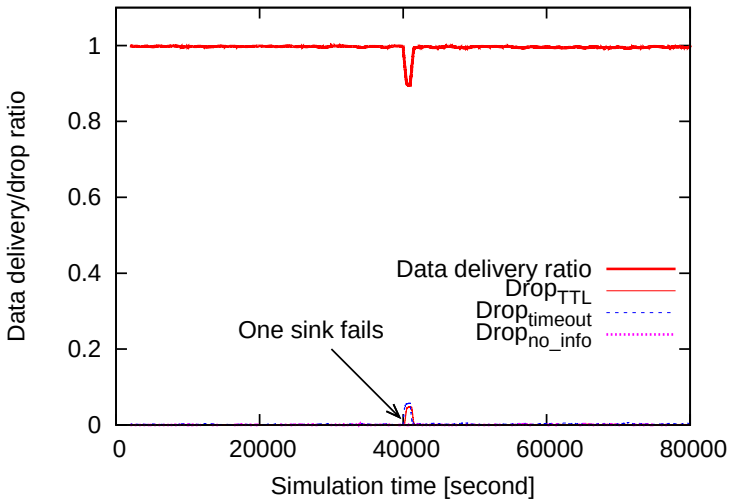


Figure 2.9: Data delivery/drop ratio in the case that sink node fails.

The reason why $\mathrm{Drop}_{\mathrm{timeout}}$ and $\mathrm{Drop}_{\mathrm{TTL}}$ increase is that the sensor node that is deployed around the failed sink node keeps sending data in upstream routing to the failed sink node until potential fields converge. When a neighbor node of the failed sink node has data in upstream routing, the node waits for an ID message from the failed sink node. Because the failed sink node cannot send an ID message, the neighbor node of the failed sink node drops the data owing to timeout.

In our method, a node updates its potential when the node receives an ID message containing a potential. Therefore, a node with data that is awake for a long time updates its potential more frequently. When a sensor node that has upstream data waits for an ID message from the failed sink node, its potential may become less than that of its neighbor node owing to frequent updates of potential. It then forwards the data to the neighbor sensor node. When the data are forwarded many times, similarly, the data are finally dropped due to the expiry of the TTL.

The result shows that PBAR is adaptive to failure of a sink node. It takes about 1,500 s for the data delivery ratio to recover to 99% after one of the sink nodes fails, which relates to the time when the potential field constructed by the failed sink is no longer used. Therefore, when each sensor node updates its potential more frequently or when a neighbor node of the failed sink node detects the failure of the sink node and broadcasts a message to no longer use the potential field constructed by the sink node, it takes less time for the data delivery ratio to recover. We denote the latter situation as situation 3 (S3) and evaluate the data delivery ratio immediately prior to 1,000 s.

Figure 2.10 shows the result for S3. The transition of the data delivery and drop ratio is similar to the result in Figure 2.9. However, in S3, $\mathrm{Drop}_{\mathrm{TTL}}$ does not increase even when a sink node fails. This is because all the upstream data are delivered to one of the three other sink nodes after sensor nodes receive information about the failure of a sink node. The time for the data delivery ratio to recover to 99% after the failure of one sink node decreases to 1,100 s.

Note that the time for the data delivery ratio to recover is much less than the time of potential convergence. This is because it is not the potential convergence itself but the gradient of the potential field that is important in potential-based routing. Therefore, the time for potential convergence does not greatly affect the data delivery ratio.

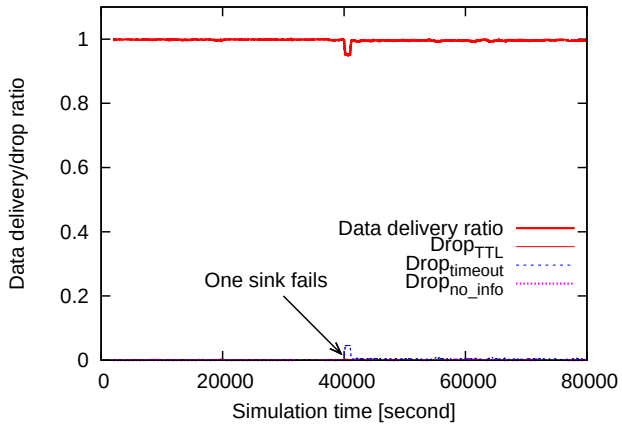


Figure 2.10: Data delivery/drop ratio in the case that sink node fails in S3.

2.7 Summary

In this chapter, we realize PBAR by merging PBUR and PBDR. In PBAR, multiple sink nodes construct independent potential fields, and all nodes have a set of potentials used as a virtual coordinate. We defined virtual distance based on virtual coordinates and use it as a routing metric. Through OMNeT++ simulation, we evaluated the data delivery ratio and path stretch for various node densities, as well as the adaptivity to failure of multiple sensor nodes or a sink node. PBAR achieves a data delivery ratio greater than 99.7% when the network has a suitable node density. Even if multiple sensor nodes fail or a sink node fails, the data delivery ratio recovers immediately after sensor node failure or sink node failure. In PBAR, when the number of potential fields increases, the reliability of any-to-any routing increases, but so does the overhead. To investigate this trade-off should be future work.

Chapter 3

Virtual Wireless Sensor Networks: Adaptive Brain-Inspired Configuration for Internet of Things Applications

3.1 Introduction

For the future Internet of Things (IoT) environment, the effective integration of various types of networks will be an important consideration. In addition to the co-existence of wired and wireless links, domains may involve sensor nodes and actuator nodes from multiple vendors. Moreover, applications running over such mixed networks will make different demands of the network, such as locational information or other node-specific information.

To realize the IoT environment, wireless sensor networks (WSNs) will be a crucial technology to allow collecting and acting on environmental information. WSNs will be integrated into future communication infrastructure [8]. Here, however, networks are considered to be constructed independently to provide service within a local area.

For such heterogeneous WSNs to be consolidated into infrastructure and to share physical sensor substrates across multiple IoT applications, virtualization of WSNs is one solution. In virtualization methods, the functionality of a WSN is split into two parts: the physical infrastructure

of sensors and the applications, which rely on the aggregated resources. The main advantage of virtualizing WSNs is that this provides the ability to realize shared infrastructure that can satisfy various service demands [32, 33].

For example, sensor substrates deployed for fire-detection or fire-tracking applications can be shared between homeowners and city administration [34]. In extant task-oriented WSNs, redundant sensor nodes are deployed in the shared domain, and the duplicated nodes are chosen according to application. This is done because the required granularity or service demands of the distinct users are different, but such redundant deployment is inefficient. Virtualization can eliminate tight coupling between applications and resources, which enables multi-application use of existing sensor substrates. Moreover, virtualization of WSNs provides a new business model, sensor-as-a-service, for both infrastructure owners and service providers [35].

Virtualization of WSNs has been studied by many researchers. The Federated Secure Sensor Network Laboratory (FRESnel) [54] and Virtualized dIstributed plaTfoRms of smart Objects (VITRO) [55] projects, in particular, are focused on the virtualization of WSNs. FRESnel aims to federate large-scale WSNs and enable the simultaneous running of multiple applications. VITRO aims to provide dynamic cooperation among sensor nodes by dividing WSNs into physical sensor substrates and applications.

In previous work on the virtualization of WSNs, virtualization at two levels has been studied: node and network [34]. Node-level virtualization methods enable a single node to concurrently process multiple applications. Approaches to node-level virtualization can be divided into three classes according to the element that provides concurrency: a virtual machine [36], the operating system [37] or middleware [38]. Under network-level virtualization, the virtual network for running a single application consists of a subset of sensor nodes. Network-level virtualization results in efficient use of resources because the nodes not used by one application can be used by applications in other virtual networks. Approaches to network-level virtualization can be divided into two classes: overlay-based solutions [13] and cluster-based solutions [11].

Within this system of classification, we aim for overlay-based network-level virtualization. Many other researchers have proposed overlay-based approaches to network-level virtualization [13, 33]. Those approaches focus mainly on providing a framework that allows the sharing of physical

sensor resources. Although improved manageability has been mentioned as a reason for virtualizing networks, the method of constructing a virtual wireless sensor network (VWSN) topology for each application in an overlay layer has not been discussed enough. Moreover, within the IoT environment, changes to traffic patterns, variation in traffic demand and the addition or removal of virtual nodes can all occur. Because of this, providing stable applications remains difficult when using the currently-existing approaches, in which only required resources (i.e., without redundancy) are assigned to a user. Reliability, even in the face of such environmental changes, is important. For that reason, we focus on a means of constructing a robust and adaptive VWSN topology for applications.

To ensure robust connectivity, each node should know all routes to the other nodes. However, keeping this information up to date would require all nodes to exchange or maintain their information with each other, resulting in heavy traffic. Especially for energy-scarce WSNs, strategies with low overhead are required. Therefore, we need to construct reliable VWSNs while keeping the overhead low for nodes. From the viewpoint of VWSN managers, the efficient distribution of sensor substrates is necessary in order to maximize per-sensor benefit. Therefore, a desirable VWSN construction method will be able to form as many virtual networks from the finite sensor substrates as possible and allow many users and applications to share the physical network. One solution to this type of problem is to use a hierarchical structure. Such structures offer both low overhead and high manageability because they can abstract the physical network and allow nodes to use their information in multiple applications.

In our previous work, we proposed a method of construction of a VWSN topology that offers robust connectivity against node removal [41]. Our approach is inspired by brain network features, and we focus particularly on the similarity between the hierarchical modular structure of brains and the modular structure that emerges from the integration of local networks.

A brain network has many structural properties, such as heavy-tailed degree distribution, rich clubs, clustering, small-world properties, hierarchical modular structure, and so on [43]. Especially, small-world properties and hierarchical modular structure contribute to the evolvability of the brain network and high communication efficiency in the global area with low metabolic cost [44]. Our conclusive goal is autonomously evolving VWSN according to the environmental changes. When

congestion or event-driven burst traffic occurs or the physical topology changes according to the addition/removal/move of nodes, the resources for the virtual topology, such as time slots for the application, the number of physical paths assigned to a virtual link or new virtual links for new physical topology, should be reassigned to meet the environment. As the first step toward this goal, we introduce the structural properties of the brain network into a VWSN topology.

Our approach consists of two steps: constructing modules with small-world properties and integrating the modules hierarchically. Our idea is a bottom-up design of the hierarchical structure, which is inspired by the brain. Due to the spatial constraints, such as the harsh wireless environment, in WSN, it can be one solution to construct a VWSN topology with taking the spatial constraints into consideration. The modules in the brain network consist of sub-modules, which have a close correlation in terms of their function. In our proposed structure, a higher-tier module consists of sub-modules that are assigned to sub-applications, and the module describes an integrated system. We expect that such a modular structure taking into account modules' functions contribute to resource-efficient solutions for users' demands. However, the evaluation of resource efficiency is out of scope here. We evaluate the topology constructed by our proposal without an autonomous configuration because the performance characteristics of the constructed VWSN topology itself are primarily important.

We investigate connection patterns within each module and between modules. We showed that the VWSN topology constructed by our proposal is robust against node removal on both connectivity and path length in our previous work [41].

In this chapter, we show an overall architecture for constructing and running VWSN services on a VWSN topology as a further investigation of our proposed method. Our main focus is providing a user a reliable VWSN network, and we show one solution providing a user IoT resources by dividing providers into infrastructural and VWSN providers. Then, we show how we connect modules to construct a reliable VWSN network. In an actual situation, the optimized topology cannot be constructed because opened information, such as technical specification, are limited by vendors. Despite of such heterogeneous environment, we need to connect networks for cooperation. Thus, we propose a policy to connect them.

Our contributions are as follows.

- Our main idea for constructing a robust VWSN topology is shown in our previous work [41]. In this chapter, we show a series of procedures for running services on the VWSN topology and demonstrate the feasibility of our proposal by simulation, including from user-level requests for a new virtualized WSN to packet-level behaviors. Considering a heterogeneous environment, we design the purpose of the infrastructure layer and the virtual layer. Then, we specify the design so as to be able to conduct a simulation evaluation. In this chapter, our main focus is on designing the virtual layer. Therefore, we solve heterogeneity of the performance of devices by defining the weight of a virtual module and a virtual link.

- We evaluate the features of our proposed method, especially adaptivity. Here, we define the adaptivity of the constructed VWSN as the ability to quickly recover its function, where various types of traffic exist. Supposing a node failure scenario, we show that our proposal can be adaptive by proactively setting up routers and reactively recovering them. The hierarchical modular structure of VWSNs allows a memory-efficient routing method with low overhead by restricting the flooding area used during route recovery. We also show that adaptivity can be low even when the VWSN network has robust connectivity.

Note that our focus is not the proposal of a routing or route recovery method itself. The routing method and the route recovery method shown in this chapter are used for clarifying the structural problem of our VWSN network that harms the reliability of the VWSN topology. Therefore, another routing method can do. Moreover, we assume that each application can use different routing strategies, and other protocols can also be configured by users according to their demands. Clearly, it should be in the virtualization scenario.

The rest of this chapter is organized as follows. Related work is shown in Section 3.2. In Section 3.3, we present a use-case scenario and give an overview of the virtualization of WSNs, which is the focus of this chapter. The description of the method that we propose for constructing a VWSN topology inspired by brain networks is shown in Section 3.4. In Section 3.5, we show a routing algorithm that combines a hierarchical modular structure with a simple method for the discovery of an alternative route after node failure. We evaluate our proposal against the use-case scenario of Section 3.7. In Section 3.8, we summarize this chapter and describe future work.

3.2 Related Work

Many researchers have worked on the virtualization of wired networks, such as the local area network (LAN) or data center. The virtual local area network (VLAN) or virtual private network (VPN) are well-known examples of the virtualization for wired networks [56]. The overlay network that has a virtual topology constructed on the physical topology is also this kind of example [57]. Constructing an overlay network can be seen as a virtual network embedding problem, which consists of virtual node mapping and virtual link mapping. Virtual nodes are allocated in physical nodes in virtual node mapping, and virtual links connecting virtual nodes are allocated in physical paths connecting the corresponding physical nodes in the virtual link mapping. In virtual network embedding, the request of constructing an overlay network is described as a graph with the required resources of virtual nodes and virtual links, such as CPU or bandwidth. In the wireless scenario, however, resource abstraction of links, such as bandwidth, is challenging because of unstable channel quality [56]. Interference and fading of wireless signals are inevitable without sophisticated time synchronization of nodes and can degrade the performance of virtual networks. Moreover, in the future IoT environment, there are various applications, and burst, periodic and time-varying traffic coexist. Therefore, it is not realistic to reserve the finite size of resources in the environment, and the existing methods for wired networks are not applicable straightforwardly.

In the topology control of a WSN, tree-based and clustering tree-based approaches are proposed for energy-efficient data collection and node-to-node routing [58]. In such a cluster-based topology, however, packets for inter-cluster communication need to go through cluster heads, which can be a single failure. In case of the failure of a cluster head, self-healing methods, such as reconstruction of the cluster [59] or reselecting of new cluster head from the candidates [60], recover connectivity of the network. Although these methods provide redundancy of connectivity, the structural properties of the network are not considered. When we consider the failure model or the attack model, we need to pay attention to the structural properties of the network. Jameii et al. applied a multi-objective optimization scheme to the topology control of a WSN [61]. Their focus is to provide a set of Pareto-optimal solutions that optimize four competitive objectives, the number of active nodes, coverage, connectivity and energy conservation, by adjusting the communication ranges of sensor nodes and

sleep scheduling of sensor nodes. In addition to the multi-objective optimization algorithm, they introduce a learning automata for the dynamical configuration of the topology according to the environmental changes. Although the simulation evaluation showed the best performance compared to the existing approaches, the discussion about routing overhead and computational overhead is lacking.

For decoupling IoT networks into the control plane and the data plane, software-defined networking (SDN) technologies are discussed. Qin et al. proposed an SDN controller for managing heterogeneous IoT networks [62]. They define a resource matching strategy between abstract task description and resource specifications and flow scheduling by using a heuristic algorithm until the algorithm finds a feasible solution for the request. Because the controller estimates the end-to-end flow performance before resource allocations, their proposal shows better throughput, delay and jitter than existing ones. Jararweh et al. proposed a software-defined IoT framework, which exploits several software defined systems, such as SDN, software-defined storage and software-defined security [63]. Their focus is to simplify the IoT management and to solve problems involved in the big data scenario. However, specific evaluations of the model were not examined. Other virtualization schemes for eliminating the tight-coupling of applications and physical substrates in IoT networks are also proposed. However, to our knowledge, how to construct the VWSN topology that supports reliable communication has not been discussed enough.

Therefore, we provide a new viewpoint to make VWSN topology reliable by introducing a brain-inspired structure, and we conduct simulation evaluation with considering environmental changes and packet-level behavior, including control-plane and data-plane packets. We also discuss the feasibility and overheads of our proposed architecture. In this chapter, however, we do not consider how to get resource efficiency or energy efficiency, which will be more critical in the virtualization scenario. Although we discuss approaching our considering world from the current real world in Section 3.7.6, how to solve these issues is out of our scope, but should be future work.

3.3 Scenario for Providing a VWSN

In this chapter, we adopt the concept of the virtualization of WSNs as shown in [32] and assume that the virtualization of WSNs is realized by two types of providers. One type is an infrastructure provider that manages physical sensor nodes and physical WSNs; the other is a VWSN provider that constructs a VWSN by aggregating physical substrates from multiple infrastructure providers. We assume that a VWSN provider is responsible for constructing a VWSN topology for an application.

In this chapter, a VWSN is provided to handle user requests as shown in Figure 3.1. A user who wants to develop a new application selects a set of source nodes and destination nodes. This set of nodes can be abstracted through an interface provided by the VWSN provider. Then, the user sends a request to construct a new VWSN, and this message is received by a VWSN provider. When the user requests abstract resources, such as the periodic report of humidity and temperature in the northeast area of Tokyo in Japan, the VWSN provider selects a set of source and destination nodes corresponding to the user's request. The request message contains the traffic pattern between the source nodes and destination nodes, as well as the required degree of reliability. The traffic patterns describe the way that data packets are generated, such as periodic or event-driven scheduling. When the VWSN provider receives the request, it checks whether there are enough resources to construct a VWSN that satisfies the request. When there are not enough resources, the VWSN provider rejects the request. Otherwise, the VWSN provider constructs a topology and assigns at least one physical path to a virtual link. When high reliability is demanded, redundant physical paths can be assigned to a virtual link. When a low number of redundant physical paths is assigned to a virtual link, the resources of sensor nodes are more fully utilized, but low redundancy may lower reliability. At this time, the VWSN provider determines which nodes the user will be allowed to use. The allowed ones include source nodes, destination nodes and relay nodes for communication between source and destination nodes. Once the set is decided, the VWSN provider calculates routing tables for each node in the constructed VWSN and sends the information, including the traffic pattern, to the gateway that connects the Internet and WSNs and may translate protocols to communicate with nodes in WSNs. The gateway sends the information to each node in the VWSN. Over time, each node in the VWSN acts as a resource for the VWSN and sends data according to the received traffic

pattern.

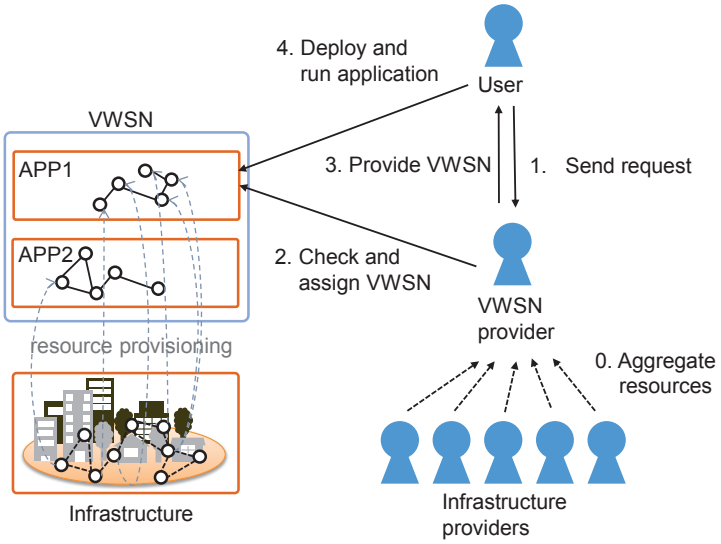


Figure 3.1: Providing a virtual wireless sensor network (VWSN) to a user.

As described in Section 3.2, it is true that it is hard to manage resources completely because of unstable channel quality and the time-varying traffic pattern. One solution for determining whether the user's request can be met or not is to limit the number of applications running on each module or node.

3.4 A Method for Configuring VWSNs to Use the Properties of Brain Networks

In this section, we show the method for constructing a VWSN topology by using the structural properties of human-brain networks. First, we describe some topological properties possessed by human-brain networks and explain the expected advantages of introducing them into WSNs in Section 3.4.1. Then, the brief explanation of the VWSN topology construction method proposed in our previous work [41] is shown in Section 3.4.2.

3.4.1 Human-Brain Networks

A brain network possesses a modular community structure and has small-world properties. The modular community structure offers robustness, adaptivity and evolvability. The small-world properties offer high communication efficiency in both local and global domains, as well as local robustness [44, 64–68].

Modular Community Structure

A brain network is spatial, which means that the length of a connection is limited by the metabolic costs associated with establishing and maintaining the connection [43, 64, 69]. As a consequence, many short-distance links are constructed and maintained in preference to long-distance links. Within the modular community structure possessed by brain networks, connections are dense and short within modules, but sparse and long between modules. Further, this type of structure is hierarchical [44, 64].

The modular structure in human-brain networks offers various advantages. The high density of connections within each module ensures robust connectivity, including many alternative routes between pairs of nodes within the same module. Moreover, connection distance and communication delay can be adjusted quickly by configuring long-distance inter-module links [64]. When cognitive demand increases, costly long-distance links are constructed so as to acquire a more efficient structure; when demand decreases, the structure becomes more highly clustered and, hence, less costly. This feature contributes most especially to evolvability.

The introduction of a hierarchical modular structure to WSNs is expected to result in a topology that can evolve adaptively to changes in resources and traffic demand. Because WSNs, like brain networks, are spatial networks, sensor nodes deployed in a close area will be densely interconnected when a hierarchical modular structure is used. Therefore, the topology of a WSN generally possesses many alternative routes, resulting in robust connectivity. Here, we construct a VWSN topology by exploiting a part of physical topology to get the robust connectivity with considering the spatial constraints of wireless communication. Moreover, the hierarchical structure is suitable for the horizontal integration of virtual networks, which increases the reusability of virtual

resources. To integrate multiple VWSNs into one VWSN, a new tier can be overlaid onto the existing VWSNs, which are then connected in the added tier. In the application that uses this integrated VWSN, each node can reuse most of the routing tables assigned by the component VWSNs.

Small-World Properties

Small-world networks are characterized by properties, such as short average path length and a high clustering coefficient. The average path length (APL) is defined as

$$\mathrm{APL} = \frac{1}{N(N-1)} \sum_{i,j} sd(i,j), \tag{3.1}$$

where N is the number of nodes and $sd(i,j)$ is the lowest hop count between node i and node j. A low APL indicates highly efficient global communication. The clustering coefficient (CC) is defined as

$$\mathrm{CC} = \frac{1}{N} \sum_{i} \frac{2e_i}{k_i(k_i-1)}, \tag{3.2}$$

where e_i denotes the number of links between neighbors of node i, and k_i is the degree of node i. The degree of a node is the number of neighboring nodes connected with the node by wireless or wired links. A high CC indicates that nodes in local regions are densely connected, which contributes to local communication efficiency.

In brain networks, densely-connected nodes within a module contribute to a high CC, which leads to efficient, segregated information processing and synchronization. Myelinated long-distance links with high electric conductivity (i.e., high reliability, high speed, long-distance links) are one contributor to global communication efficiency in brain networks [64]. The short communication delay of this type of link enables close cooperation between different regions.

In the IoT environment, we can consider that the amount of traffic between a pair of nodes located at short distance is larger than that between a pair of nodes located at large distance, which is similar traffic demands to that of the brain network. Thus, small-world properties will handle IoT traffic effectively. To model this property in a VWSN topology, we first connect modules located

in shorter distance and introduce a virtual long-distance connection that connects two physically-distant sensor nodes. This leads to a reduction in communication delays across the whole network because it reduces the APL while many parts of the network keep their overheads low.

3.4.2 VWSN Topology Construction Method

As discussed above, a topology with the small-world properties will have high global communication efficiency, and a highly modular topology will have efficient segregated information processing and robust connectivity. We previously proposed a means of constructing a VWSN topology with small-world properties and high modularity [41]. Here, we integrate modules hierarchically to construct a VWSN topology. At the same time, we add long-distance links to a clustered topology to give the VWSN topology the small-world properties at each tier. Figure 3.2 shows an example of hierarchical VWSN construction. The first tier of the VWSN is a network in the minimum-unit module. The second-tier VWSN is constructed by integrating unit modules. The third-tier VWSN is constructed by integrating virtual sensor networks deployed for sub-applications.

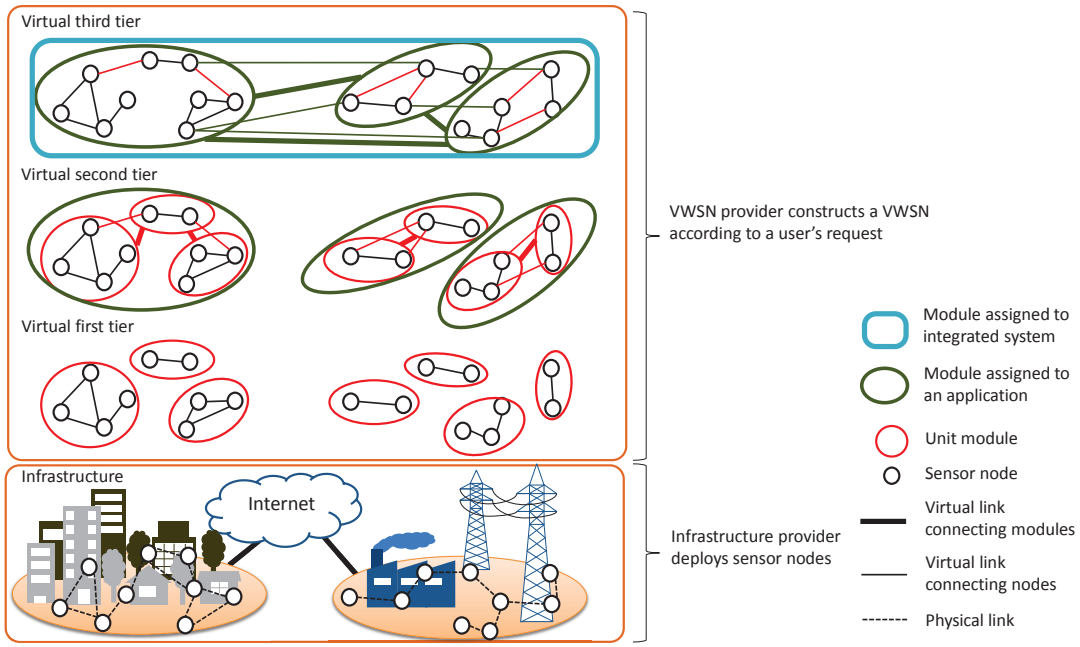


Figure 3.2: Example of a hierarchical VWSN topology.

In this section, we briefly explain the method proposed in [41]. We assume that adding multi-hop wireless or wired links contributes to reachability between any two nodes. After this, a virtual link is mapped to the shortest physical path in an infrastructural topology. Note that we consider that the physical topology of WSN is the unit disk model and does not have small-world properties. We construct a virtual network having the small-world properties that contribute to a short average path length by embedding the small number of long-distance virtual links.

First, a VWSN provider defines a set of nodes for a user application. The constructed VWSN must contain a set of source and destination nodes requested by the user, a set of relay nodes for communication between source and destination nodes and redundant nodes according to the user's demand for reliability. When these sets of nodes have been determined, the VWSN provider selects a set of modules to construct a VWSN and integrates them by embedding virtual links between them.

Construction of an L-th-tier VWSN topology can be divided into two smaller subproblems. In the first problem, we regard each $(L-1)$-th-tier VWSN as one module (M_i^{L-1}) in the L-th tier, where i indicates the specific module, and the choice to be made is how to connect pairs of modules. In the second problem, sensor nodes are to be mapped to the endpoints of L-th-tier virtual links. In this, M_*^0 denotes a sensor node and M_*^1 denotes a module clustered by applying the Newman algorithm [70].

For the first problem, we construct an L-th-tier virtual topology by adding virtual links between $(L-1)$-th-tier modules. First, an initial virtual topology is constructed: when two nodes belonging to different $(L-1)$-th-tier modules are connected by a physical link, an $(L-1)$-th-tier virtual link is embedded between these $(L-1)$-th-tier modules. Second, new $(L-1)$-th-tier virtual links are embedded into the initial virtual topology according to a preferential attachment rule. The probability of adding an $(L-1)$-th-tier virtual link between M_i^{L-1} and M_j^{L-1} is

$$p_{\text{intra}}^L(M_i^{L-1}, M_j^{L-1}) = \frac{\frac{G^{intra}(k_{M_i^{L-1}}, k_{M_j^{L-1}})}{F(h_{M_i^{L-1}, M_j^{L-1}})}}{\sum_{e_{M_a^{L-1}, M_b^{L-1}} \in \bar{E}_0^L} \frac{G^{intra}(k_{M_a^{L-1}}, k_{M_b^{L-1}})}{F(h_{M_a^{L-1}, M_b^{L-1}})}}. \tag{3.3}$$

Here, $(L-1)$-th-tier modules M_i^{L-1} and M_j^{L-1} belong to the same L-th-tier module, $\bar{E}_0^L$ is the set of virtual links in the graph complement of the L-th-tier initial virtual topology and F is a cost function $F(d) = \mathrm{e}^{d/d_c}$, where d_c is a constant parameter characterizing distance constraints. $h_{M_i^{L-1},M_j^{L-1}}$ denotes the minimum hop count between M_i^{L-1} and M_j^{L-1} in the L-th-tier initial virtual topology, and $k_{M_i^{L-1}}$ denotes the degree of M_i^{L-1}. The degree of an L-th-tier module is the number of neighboring L-th-tier modules connected with the module by L-th-tier virtual links. Strategy function G^{intra} is a function for embedding a new link preferentially according to the degrees of the endpoint modules. The strategy $intra$ for adding a new link can be any of "high-high (hh)," "low-low (ll)" and "high-low (hl)." When $intra = \mathrm{hh}$, a pair of higher degree modules is selected preferentially for a new link, and when $intra = \mathrm{ll}$, a pair of lower degree modules is selected for a new link. When $intra = \mathrm{hl}$, a higher degree module and another lower degree module are selected preferentially and connected. The definitions of G^{intra} for each possibility are as follows:

$$\begin{aligned} G^{hh}(k_i, k_j) &= k_i \cdot k_j \\ G^{ll}(k_i, k_j) &= k_i^{-1} \cdot k_j^{-1} \\ G^{hl}(k_i, k_j) &= \max(k_i, k_j) \cdot |k_i - k_j| \end{aligned}$$

The number of added L-th-tier virtual links is $\left\lceil C^L_{\mathrm{intra}} |E_0^L| \right\rceil$, where $|E_0^L|$ is the number of links embedded in the L-th-tier initial virtual topology and C^L_{intra} is a constant satisfying $0 < C^L_{\mathrm{intra}} \leq 1$.

For the second problem, we describe a method for mapping the endpoints of an L-th-tier virtual link to sensor nodes. In this method, we recursively select the endpoints of an L-th-tier virtual link from its submodules until the endpoint nodes are determined. We define the probability $p^L_{\mathrm{inter}}(M_i^{L-1}, M_j^{L-1})$ of mapping an L-th-tier virtual link to an $(L-1)$-th-tier virtual link between M_i^{L-1} and M_j^{L-1} in the same way as Equation (3.3). The strategy $inter$ for mapping can be one of "High-High (HH)," "Low-Low (LL)" and "High-Low (HL)," with the same meanings as "hh," "ll" and "hl," respectively, but applied to links between modules. The number of virtual links for each mapping is $\left\lceil C^L_{\mathrm{inter}}(\left|E^{M_x^L}\right| + \left|E^{M_y^L}\right|) \right\rceil$, where $\left|E^{M_x^L}\right|$ denotes the number of $(L-1)$-th-tier virtual links in the L-th-tier VWSN topology of M_x^L, and C^L_{inter} is a constant value satisfying

$0 < C^L_{\text{inter}} \leq 1$. We call an endpoint node of a virtual inter-module link a connected node.

Next, we assign a virtual link to one of physical shortest paths, and the routing tables are constructed by the method mentioned in the following section. We consider that traffic within a module can be routed along the shortest path. However, because we abstract paths between modules, traffic between modules cannot always be routed along the shortest path. To improve throughput, we introduce small world properties to a VWSN topology. Note that actual throughput depends on a means of assigning a virtual link to physical paths. Thus, in our architecture, different policies for assigning a virtual link to physical paths can be set to each service as each user requests. The VWSN topology constructed by our proposal has the risk that a physical link shared by multiple services can be a bottleneck. However, our proposed design includes the assignment of a virtual link to multiple physical paths, which is the part to tackle the problem.

From here, we suppose that any node can be reassigned to a relay node of a physical path of any virtual link in the event of node failure. Therefore, we assume that the user requires the maximum degree of reliability. The route recovery method is also described in the following section.

3.5 Routing over the Hierarchical Virtual Topology

In a routing protocol for a hierarchical topology, a within-module network can be aggregated into an abstract topology. Owing to this abstraction of modules, it is not necessary for each sensor node to hold information about every path to each node. This improves the efficiency of memory usage. To take advantage of this, we use minimum-weight path routing, considering the modular structure of a VWSN topology.

In Section 3.5.1, we describe an overview of the routing algorithm we use. In Section 3.5.2, we show the routing tables that each node needs to hold for deciding a forwarding node, and in Section 3.5.3, we show a path recovery method. We show the details of the packet format and the definition of path weight used in our simulation evaluations in Appendices 3.A and 3.B, respectively.

3.5.1 Overview

In this section, we give an overview of our routing algorithm. Figure 3.3 shows an example of a three-tiered virtual topology. Each node belongs to one module in each tier. For example, node a belongs to module M_C^1 in the first tier, module M_A^2 in the second tier and module M_A^3 in the third tier. The end nodes of each higher-tier link are assigned to physical nodes. In Figure 3.3, the end nodes of the first-tier link (M_A^1,M_C^1) are assigned to nodes g and c, and the end nodes of the second-tier virtual link (M_A^2,M_B^2) are assigned to nodes k and m. In this chapter, as mentioned in Section 3.3, the VWSN provider calculates the routing tables and sends them along with the traffic patterns to each node. At this time, the VWSN provider informs source nodes of which modules the corresponding destination node belongs to in each tier. Then, source nodes can refer to the information when sending a data packet to their destination nodes.

Below, we show a method for choosing a next-hop node, which we call a forwarding node, when node s sends a data packet destined for node d.

1. Node s checks the highest tier (L) in which nodes s and d belong to different modules. Then, node s sets the initial destination module as the L-th-tier module to which node d belongs.

2. Descending from tier L, node s chooses a forwarding module in each tier as follows.

   (a) Node s chooses a neighboring L-th-tier forwarding module (M_f^L) toward the L-th-tier destination module according to the path weight.

   (b) Node s chooses an $(L-1)$-th-tier module (M_a^{L-1}) that belongs to the same module as node s in the L-th tier and is connected to an $(L-1)$-th-tier module (M_b^{L-1}) belonging to M_f^L.

   (c) Node s sets the destination module to M_a^{L-1} when node s does not belong to M_a^{L-1}. Otherwise, node s sets the destination module to M_b^{L-1}, which is connected to M_a^{L-1} and belongs to M_f^L.

3. By iterating Process 2 until L becomes zero, node s can find a forwarding node.

In this example, node s and node d belong to the module, M_A^3, in the third tier. Therefore, node s chooses a forwarding module from the second tier. At first, because module M_A^2 and M_B^2

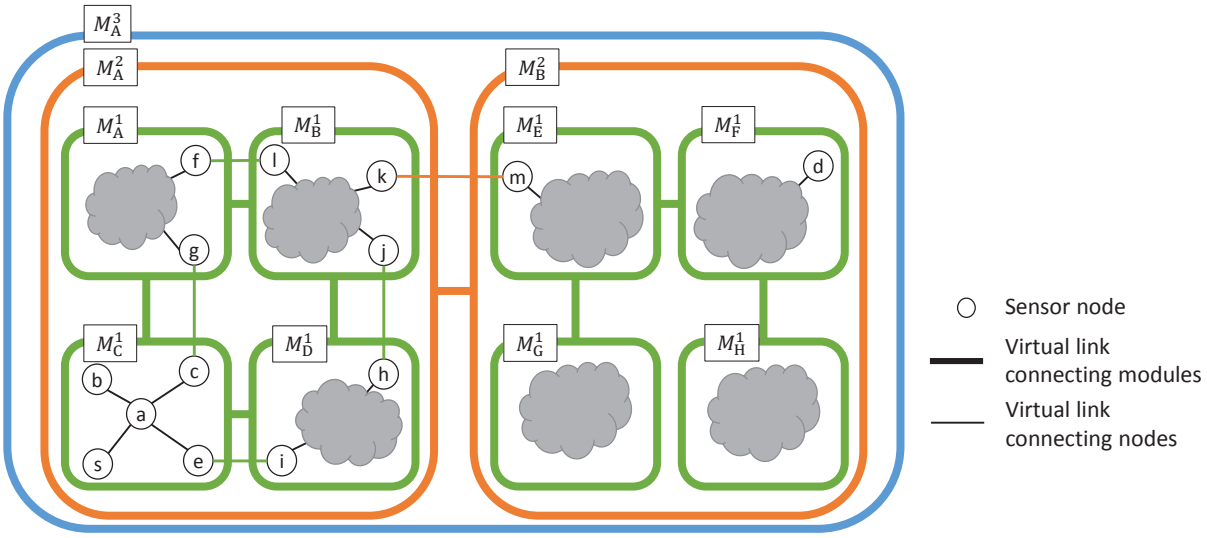


Figure 3.3: Example of a hierarchical topology.

are connected directly in the second tier, a forwarding module in the second tier is M_B^2. The first-tier modules at the end of the second-tier virtual link (M_A^2,M_B^2) are assigned to M_B^1 and M_E^1. Therefore, the data packet needs to go through module M_B^1. Then, node s needs to choose a first-tier forwarding module to deliver the data packet to module M_B^1. There are two candidates for the path to module M_B^1, and node s selects one of them according to path weight. When node s chooses module M_D^1 as the first-tier forwarding module, the data packet needs to go through node e because the end nodes of the first-tier virtual link (M_C^1,M_D^1) are assigned to nodes e and i, respectively. Finally, node s chooses node a as a forwarding node and sends the data packet to it to deliver the data packet to node e. By the same procedure, each node that receives a data packet chooses a forwarding node.

3.5.2 Routing Tables

In this section, we describe the tables that need to be managed by each sensor node for forwarding the data packets. To realize the routing algorithm mentioned in Section 3.5.1, three types of tables are needed. One is a routing table used to decide a forwarding module from neighbors in each

tier; another is a connection table used to manage the $(L-1)$-th-tier module identifiers, which are assigned to the endpoint modules of L-th-tier virtual links; the last is a long-link table used to choose a forwarding node to be assigned to a path for a virtual long-distance link. Each node requires L routing tables, $(L-1)$ connection tables and at most one long-link table in order to determine a forwarding node, where L is the number of tiers to which the node belongs. Nodes not belonging to any physical paths that are part of a virtual long-distance link do not need a long-link table.

The L-th-tier routing table is constructed for routing between $(L-1)$-th-tier modules belonging to the same L-th-tier module. When node n belongs to the L-th-tier virtual module M_A^L, it records entries of all of the $(L-1)$-th-tier modules belonging to M_A^L in its L-th-tier routing table. As a particular consequence, this permits routing between any pair of nodes belonging to the same first-tier module by using the zeroth-tier routing table.

The L-th-tier connection table converts L-th-tier forwarding module identifiers to the $(L-1)$-th-tier module identifiers associated with the $(L-1)$-th-tier module belonging to the L-th-tier forwarding module. Thus, when we talk about a "connected module" of the virtual module M_A^L, we mean an $(L-1)$-th-tier virtual module that belongs to the L-th-tier virtual module M_A^L and is connected via an $(L-1)$-th-tier virtual link to an $(L-1)$-th-tier virtual module belonging to a different L-th-tier virtual module. Because there may be more than one $(L-1)$-th-tier connected module pointing toward a single L-th-tier virtual module, multiple entries for a single L-th-tier virtual module are allowed in the L-th-tier connection table.

The long-link table is used for routing between nodes that are connected by a virtual link, but cannot communicate directly with each other due to the distance between them. We assign a physical multi-hop path to this kind of virtual long-distance link. Then, each node along the assigned path holds a long-link table for use in choosing a forwarding node toward the destination end of the virtual long-distance link.

3.5.3 Path Recovery after Node Failures

When a node needs to find an alternative path because of the failure of a neighbor node, it is necessary to separately consider two cases: discovery of the minimum-weight path between two nodes that belong to the same first-tier module and recovery of a virtual long-distance link. Moreover, recovery of a virtual long-distance link can be considered for two different cases, as follows.

1. Failure of the relaying node of a path assigned to an L-th-tier virtual long-distance link $(0 \leq L)$.

2. Failure of the end node of an L-th-tier virtual long-distance link $(0 \leq L)$.

In Situation 2, there is no way of selecting an alternative end node for recovery of the virtual long-distance link. By assigning some redundant nodes as the end node of a virtual link, the virtual link can be recovered. In this chapter, however, the node does not recover the virtual long-distance link in this situation, and the virtual long-distance link is lost, because we want to evaluate the adaptivity of the initial virtual topology. Therefore, we describe a method for the discovery of an alternative path in only Situation 1 for the recovery of a virtual long-distance link.

The methods are based on finding a reverse path by a flooding control message. To detect the failure of a neighboring node, each node periodically broadcasts a Hello packet and constructs a table of neighbors. When a node detects the failure of a neighboring node via the absence of that node's Hello packet, it will flood a control message when it is necessary to find an alternative path. In some cases, nodes in the VWSN need to send many Hello packets. Meanwhile, there is a case that nodes detect the failure of a node only when the end-to-end communication failed. Although managing devices is a crucial viewpoint of network design, our scope is to show the adaptivity of the VWSN topology, and a method of resource management is out of scope of this thesis.

We note that the flooding range of control packets can be restricted in a first-tier module for the discovery of the alternative path between two nodes that belong to the same first-tier module. This reduces the number of control packets involved in forwarding. In contrast, in the recovery of a path crossing some modules, an alternative path may not be able to be found without wide-range flooding.

Additionally, we need to consider a method that can discover a route between two L-th-tier modules when all of the virtual links between them have been lost. The details of the method for this are shown in Section 3.5.3.

Path Recovery between Two Nodes Belonging to the Same First-Tier Module

Each node periodically broadcasts a Hello packet. When node n does not receive a Hello packet from node i for a certain time, node n considers node i to have failed. Then, node n needs to find an alternative path for any path whose forwarding node is node i.

For this, node n floods a route request packet to each destination node whose forwarding node from node n is node i. After a fixed time T_{reply} since the destination node d has received the first route request packet from node n, it chooses the route request packet with the shortest path and sends a route reply packet to node n along the reverse path contained in that packet. Each node on the reverse path updates its zeroth-tier routing table when it receives the route reply packet. Note that the flooding range of the route request packet can be restricted within the module during this path recovery.

Recovery of a Zeroth-Tier Virtual Long-Distance Link

When recovering a zeroth-tier virtual long-distance link, the flooding range of control packets can be restricted to a first-tier module. Therefore, a zeroth-tier virtual long-distance link can be reassigned to an alternative shortest physical path with low overhead.

When a node assigned as a relay node in a zeroth-tier virtual long-distance link (s, d) detects the failure of the forwarding node along (s, d), it sends an error detection packet to source node s of the virtual long-distance link, via the reverse path of (s, d). After node s receives the error detection packet, it carries out the same route discovery method mentioned in Section 3.5.3. The difference from that method alone is that the table to be updated is the long-link table.

Recovery of an L-th-Tier Virtual Long-Distance Link

For recovery of an L-th-tier virtual long-distance link, wide flooding may be necessary because a physical path assigned to an L-th-tier virtual long-distance link may traverse many first-tier modules. Therefore, we adopt a patching method for the recovery of an L-th-tier virtual long-distance link as a way to reduce overhead. The obtained path may not be the shortest path, because we do not find an end-to-end path; however, the recovery time and the overhead can be reduced by not insisting on the optimal path.

In recovering the L-th-tier virtual long-distance link (s, d) assigned to path $[s\ (= r_0), r_1, r_2, \cdots, r_{l-2}, d\ (= r_{l-1})]$ from the failure of node r_u $(0 < u < l-1)$, each node r_v $(0 \leq v \leq u-1)$ floods a route request packet and searches a path to each node r_w where $(u+1 \leq w \leq l-1)$. Here, the range of flooding is restricted by the time-to-live (TTL) of the route request packets.

Path Recovery between Two L-th-Tier Modules

We suppose that a virtual long-distance link is lost when an end node of the virtual long-distance link fails or the virtual long-distance link cannot be recovered within a certain time. When all physical paths assigned to an L-th-tier $(2 \leq L)$ virtual long-distance link are lost, the L-th-tier routing tables should be updated to route a data packet in the L-th tier. Although the principal idea is the same as that of the method shown in Section 3.5.3, flooding in an L-th-tier virtual network is different from general flooding. We call this modified form L-th-tier flooding.

L-th-tier flooding can be realized by multicasting among only connected nodes, because a connected node list corresponding to an L-th-tier module list is required for path recovery between two L-th-tier modules. When the connected node tries to find an alternative route between L-th-tier modules, it sends an upper route request packet to the other connected nodes belonging to its own first-tier module. Then, that node and each of the connected nodes that received an upper route request packet sends the upper route request packet to all connected nodes that belong to neighboring first-tier modules. When a connected node belonging to the destination module receives an upper route request packet, it sends an upper route reply packet to the sender of the upper route request.

3.6 Simulation Evaluation of the Robustness of VWSN

We evaluate our proposed method and compare it with a bio-inspired small-world network construction method (we call it bio-inspired) [71]. We briefly explain this method in Section 3.6.1. We call our proposed method the brain-inspired configuring method (BICM). Because the BICM method characterizes nine distinct configuration patterns, according to the choice of a combination of $intra$ and $inter$, we identify the specific pattern by BICM($intra$,$inter$).

3.6.1 Bio-Inspired Techniques for Achieving Small-World Properties

The target of the bio-inspired method is WSNs with non-uniform node density. To achieve small-world properties in such WSNs, the method uses bio-inspired techniques [71]. We regard the constructed topology as a virtual topology, although this method is not for constructing a virtual topology. The bio-inspired method consists of two steps: clustering by using a lateral inhibition technique and identifying nodes for constructing long-distance links by using a flocking technique. After the clustering process, all nodes are associated with the maximum-degree cluster head within η hops, where η restricts the maximum hop distance of the cluster. In the bio-inspired method, a long-distance link is embedded between a peripheral and a centroid node of a cluster for the efficient reduction of average path length. A centroid node is a node with the maximum closeness centrality among nodes in the cluster, and a peripheral node is a node that is located at the boundary of the cluster. Each peripheral node randomly selects the beam length, subject to the restriction by the maximum antenna elements Φ. Then, it looks for centroid nodes within range of a beam of the selected length and nominates potential endpoints of a long-distance link. Each centroid node that is already connected to a neighboring peripheral node is excluded from the set of candidates. Finally, a long-distance link is constructed to the candidate centroid node possessing the highest minimum hop count from the peripheral node.

3.6.2 Evaluation Metrics

We evaluate a VWSN topology in terms of small-worldness, average path length in the virtual network, average path length in the physical network, total number of virtual links, modularity,

robustness of connectivity, and robustness of average path length.

In [72], the metric ω which described small-worldness of topology was proposed. ω is calculated from the clustering coefficient, that of an equivalent lattice network, average path length, and that in an equivalent random network. For this purpose, equivalence between networks indicates that they have the same degree distribution. Formally, ω is defined as

$$\omega = \frac{L_{rand}}{L} - \frac{C}{C_{latt}}, \tag{3.4}$$

where L and L_{rand} are average path length of the original network and equivalent random network respectively; C and C_{latt} are the clustering coefficient of the original network and an equivalent lattice network respectively. The value of ω is in the range of $[-1, 1]$. When $\omega \simeq 0$, the original network has small-world properties; when $\omega \simeq 1$ it has random-like properties; and when $\omega \simeq -1$ it has lattice-like properties.

We define two types of average path length, denoted by APL, APL in the virtual network (vAPL) and APL in the physical network (pAPL). The value of vAPL is APL when nodes connected by a virtual link can communicate with each other directly. The value of pAPL is APL when nodes connected by a virtual link communicate with each other via the shortest multi-hop path in the physical network. Actual APL in physical networks may change depending on a means of realizing a long-distance link in the physical network. Thus, vAPL suggests the minimum APL and pAPL suggests the maximum APL in a VWSN.

In [70], the metric Q which describes modularity was proposed. The definition of modularity is the following:

$$Q = \sum_{i} (e_{ii} - a_{ii}^2), \tag{3.5}$$

where i denotes a group identifier and e_{ii} denotes the ratio of the number of links whose endpoints belong to the same group to the total number of edges; a_{ii} is the probability that at least one of the endpoints of an uniformly randomly chosen link belongs to group i. Then, a_{ii}^2 is the expected probability that both endpoints of a link belong to the same group.

The robustness of connectivity and APL are evaluated by removing nodes one at a time. We

evaluate the robustness of connectivity by comparing the decrease in component size which describes the number of nodes belonging to the maximally connected subgraph; we evaluate the robustness of APL by comparing the increase in APL. When all paths between node i and node j are lost owing to the removal of nodes, APL is calculated by looking on the hop count between them as the number of nodes. We suppose two modes of node removal: random failure and targeted attack. In the random failure mode, we randomly choose a node to be removed in the next time step. In the targeted attack mode, we choose the highest degree node to be removed in the next time step.

3.6.3 Evaluation Environments

In this chapter, we define that reliability of a VWSN topology consisting of its robustness and adaptivity. The robustness of a VWSN topology, which is defined as the ability to keep its connectivity high and path length short even when nodes are removed. Here, we evaluate the robustness of a VWSN topology constructed from the network, which is composed of two sensor networks and one wired link. Each of the sensor networks consists of 150 sensor nodes deployed in a domain of size $1000 \times 1000\ \mathrm{m}^2$. For one of the sensor networks, we deploy 150 sensor nodes at randomly-selected positions in the rectangular area with corners, denoted in meters along the coordinate axes of the domain, at $(0, 0)$ and $(400, 1000)$; we deploy the other 150 sensor nodes at randomly-selected positions in the rectangular area given by $(600, 0)$ and $(1000, 1000)$. Additionally, one wired link connects the two sensor networks, and its endpoint nodes are static once they are chosen. We assume the wireless communication range is 100 m.

In this simulation, we construct a three-tiered VWSN topology. We use OMNeT++ [52] to perform the simulation experiments, and the parameter settings are shown in Table 3.1. When the physical topology described above is used, the value of $\left|E^{M_x^1}\right| + \left|E^{M_y^1}\right|$ is comparatively high, which means that many virtual links are added between first-tier modules; this results in low modularity. To correct for this, we set C_{inter}^1 to a value lower than C_{inter}^L in the higher tier.

Table 3.1: Parameter settings. BICM, brain-inspired configuring method.

Method	Parameter	Value
BICM	C^L_{intra}	0.1
	$C^L_{\text{inter}}(L \neq 1)$	0.1
	C^1_{inter}	0.01
Bio-inspired	η	4
	Φ	6

3.6.4 Structural Properties

In this section, we evaluate structural properties of a VWSN topology and summarize, in Table 3.2, its small-worldness ω, vAPL, pAPL, total number of virtual links and modularity Q.

Although a BICM-based VWSN possesses small-world properties, it is comparatively lattice-like. In BICM, a means of selecting the endpoints of the inter-module links has strong effects upon vAPL and pAPL. In case of $inter = \text{HH}$ or $inter = \text{HL}$, the vAPL of an entire network is relatively small because long-distance links are embedded between nodes with high degree. In contrast, vAPL of an entire network is relatively large in case of $inter = \text{LL}$. A bio-inspired VWSN shows the most small-world properties because ω approximately equals zero. Moreover, though the number of virtual links is the largest, it achieves the smallest vAPL and pAPL among all the methods because of the flocking technique. Because peripheral nodes do not connect to centroid nodes to which its neighbor has already been connected, the long-distance links are dispersed all around the network. Moreover, APL is drastically reduced because a peripheral node chooses the centroid node to which the shortest hop count is the largest as an endpoint of a long-distance link.

3.6.5 Robustness of Connectivity

The robustness of connectivity in regard to random failure and targeted attack are evaluated in this section. Figures 3.4(a) and 3.4(b) show the decrease in component size when removal modes are set to random failure and targeted attack, respectively. Note that the probability that the nodes at the end of a wired link fail is smaller than the other nodes because such nodes can charge their batteries through the wired link. From this, we suppose that the nodes at the end of a wired link do not fail.

The decrease in component size is almost the same for each method in case of random failure.

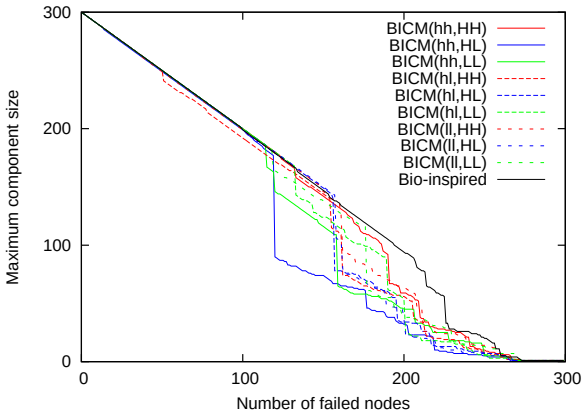
Table 3.2: Comparison of VWSNs constructed by each method

	ω	vAPL	pAPL	# of virtual links	Q
BICM(hh,HH)	-0.419	4.56	10.65	1576	0.829
BICM(hh,LL)	-0.491	5.28	13.47	1576	0.834
BICM(hh,HL)	-0.428	4.69	10.86	1577	0.831
BICM(ll,HH)	-0.403	4.71	10.11	1578	0.839
BICM(ll,LL)	-0.434	5.14	11.25	1576	0.842
BICM(ll,HL)	-0.365	4.46	11.01	1577	0.834
BICM(hl,HH)	-0.400	4.42	10.35	1578	0.834
BICM(hl,LL)	-0.410	4.80	11.74	1579	0.806
BICM(hl,HL)	-0.400	4.63	10.98	1577	0.833
Bio-inspired	-0.163	3.57	8.13	1683	0.730

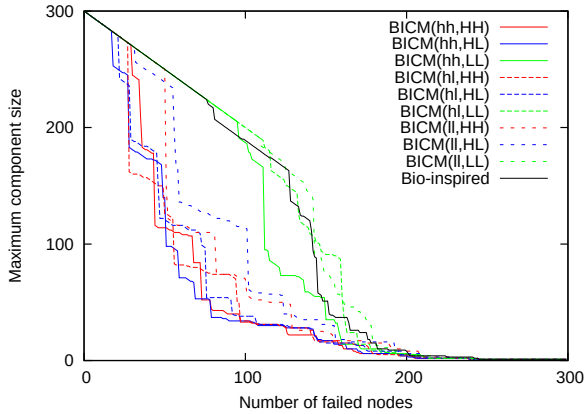
In BICM, when removal mode is set to targeted attack, links between modules are removed with high probability, which may result in sharp decrease of component size. In case of $inter = \mathrm{LL}$, a VWSN has high robustness of connectivity because it remains the component sizes high. A VWSN constructed by the bio-inspired method also has high robustness of connectivity because the whole physical topology remains in its virtual topology.

3.6.6 Robustness of Average Path Length

The robustness of vAPL and pAPL in regard to random failure and targeted attack are evaluated in this section. Figures 3.5(a) and 3.5(b) show the tendency toward the increase in vAPL when removal modes are set to random failure and targeted attack, respectively. In Figure 3.5(a), sharp increase of vAPL is caused by the failure of an endpoint node of an inter-module link and the magnitude of such a jump of vAPL describes the impact of a node failure. This jump can be seen at any results of BICM and the bio-inspired method. It is noteworthy that the failure of one endpoint node of an inter-module link can cause the sharp increase of vAPL when using strategy of $inter = \mathrm{HH}$ or $inter = \mathrm{HL}$. This is because long-distance links are concentrated to a small fraction of endpoint nodes of an inter-module link. On the other hand, the VWSN constructed by BICM with strategy $inter = \mathrm{LL}$ is robust since long-distance links are decentralized. When we construct a VWSN by bio-inspired method, a pair of a centroid node and a peripheral node



(a) Against random failure



(b) Against targeted attack

Figure 3.4: Robustness of connectivity

is connected by a long-distance link. This means that two modules are likely to be connected by two or more long-distance links, between a centroid node and a peripheral node. Therefore, vAPL increases sharply in bio-inspired method when several centroid nodes are failed. Because vAPL does not increase sharply until multiple centroid nodes fail, bio-inspired VWSN is robust of vAPL. In targeted attack, however, a BICM-based VWSN with the strategy $inter = \mathrm{HH}$ or $inter = \mathrm{HL}$, or bio-inspired-based VWSN, is vulnerable on vAPL. This is because a node with high degree which is connected by inter module link is removed at an early step. When removal mode is set to targeted attack, a BICM-based VWSN with strategy $inter = \mathrm{LL}$ is highly robust in terms of vAPL.

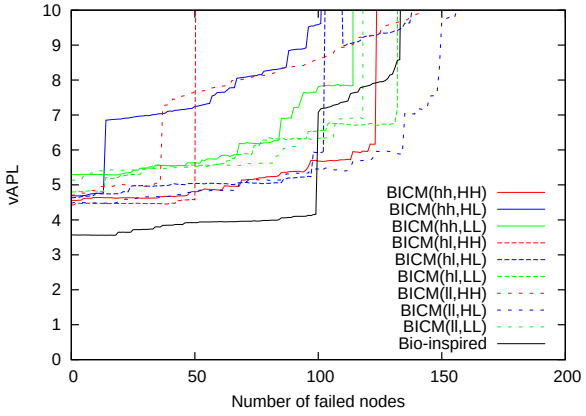
Figures 3.6(a) and 3.6(b) show the tendency toward increase in pAPL when removal mode is random failure and targeted attack, respectively. Robustness of pAPL of a VWSN constructed by respective method is on the same level as robustness of vAPL against random failure. A VWSN constructed by the bio-inspired method has a lower pAPL because its topology is almost the same as those of the physical network and almost all the physical shortest paths are available. On the other hand, since communication between modules is allowed only via nodes selected as endpoints of virtual links, extra hops caused by detour are more common in our proposal. When removal mode is targeted attack, a BICM-based VWSN with the strategy $inter = \mathrm{LL}$ is highly robust in terms of pAPL. A bio-inspired VWSN has the highest robustness on pAPL in regard to targeted attack because its topology is the almost same as that of physical networks. From the above, BICM with the strategy $inter = \mathrm{LL}$ is the method which achieves high robustness with regard to both vAPL and pAPL.

When we consider all results, multi-tiered VWSNs constructed by our proposed method have small-worldness, communication efficiency, robustness of connectivity, and robustness of APL. The evaluation of two-tiered VWSN topology give the same results as shown above. This suggests that the sub-networks observed at an arbitrary tier (scale level) of the VWSN constructed by our method will have similar properties.

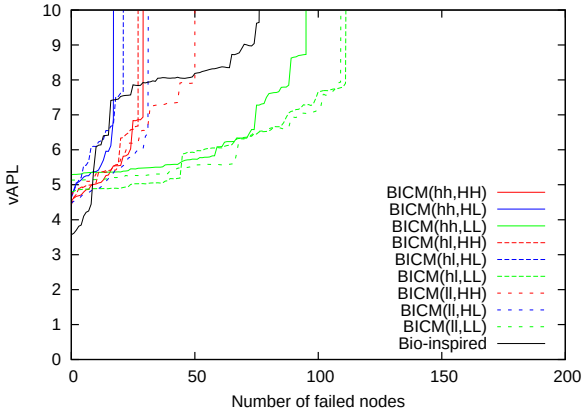
3.7 Simulation Evaluation of the Dynamic VWSN

In above section, we showed how we connect the modules to construct a VWSN network having robust connectivity and a robust path length against nodes' failure. However, the robust connectivity does not guarantee the reachability of the network. Because the congestion or burst traffics can lead to the loss or inconsistency of table information, robust connectivity is not enough for reliable communication on the VWSN network. Therefore, we show the adaptivity involved in the VWSN network and how to connect modules to construct a reliable VWSN network.

In consideration of the results shown above section, we evaluate the VWSN topology constructed by BICM(ll,LL), which is one of the most robust ones against node removal. For comparison, we also evaluate the VWSN topologies constructed by BICM(hh,HH) and by the bio-inspired



(a) Against random failure



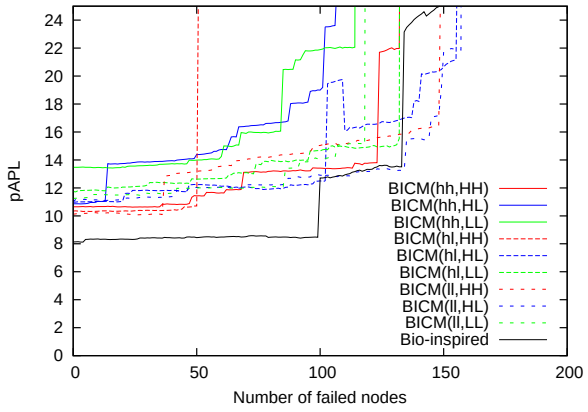
(b) Against targeted attack

Figure 3.5: Robustness of average path length in the virtual network (vAPL)

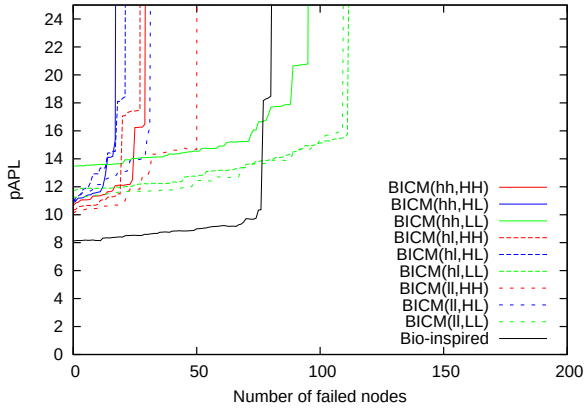
small-world network construction method (the bio-inspired method) [71]. We show that the VWSN topology constructed by BICM(hh,HH) is one of the most vulnerable to node removal in the targeted attack mode of the nine strategies. Therefore, we expect that the adaptivity of the VWSN topology of BICM(hh,HH) is also low.

3.7.1 Evaluation Metrics

Here, we evaluate the adaptivity of a VWSN topology. In this chapter, we define adaptivity as the ability to quickly find an alternative path after node failure. To evaluate the adaptivity of the VWSN topology, we evaluate the time ($T_{recovery}$) needed for the data delivery ratio to recover to at least



(a) Against random failure



(b) Against targeted attack

Figure 3.6: Robustness of average path length in the physical network (pAPL)

99.9% after the first sensor node fails. We also evaluate the number of control packets that were sent to find an alternative path ($CP_{recovery}$) after node failure. We evaluate the data delivery ratio from $t - 500$ (in s) to t, where t is an arbitrary time in the simulation. As we did in our previous paper, we assume two removal modes: random failure and targeted attack. The node to be removed at the next time step is selected randomly in random failure mode, and the node with the highest degree is selected in targeted attack mode.

3.7.2 Evaluation Environments

We evaluate the adaptivity of a VWSN topology constructed from the network, which is composed of two sensor networks and one wired link as described in Section 3.6.3. We assume the wireless communication range is 100 m. In this simulation, we construct a three-tiered VWSN topology. We use OMNeT++ [52] to perform the simulation experiments, and the parameter settings are shown in Table 3.1.

The flow of simulation is described below.

1. The VWSN provider receives, from a user, a request to construct a new VWSN topology and data on traffic patterns.

2. The VWSN provider constructs a VWSN topology and calculates the routing tables, connection tables and long-link tables for each node.

3. The VWSN provider informs the gateway about the tables of each node and the traffic pattern.

4. The gateway sends the information to each node.

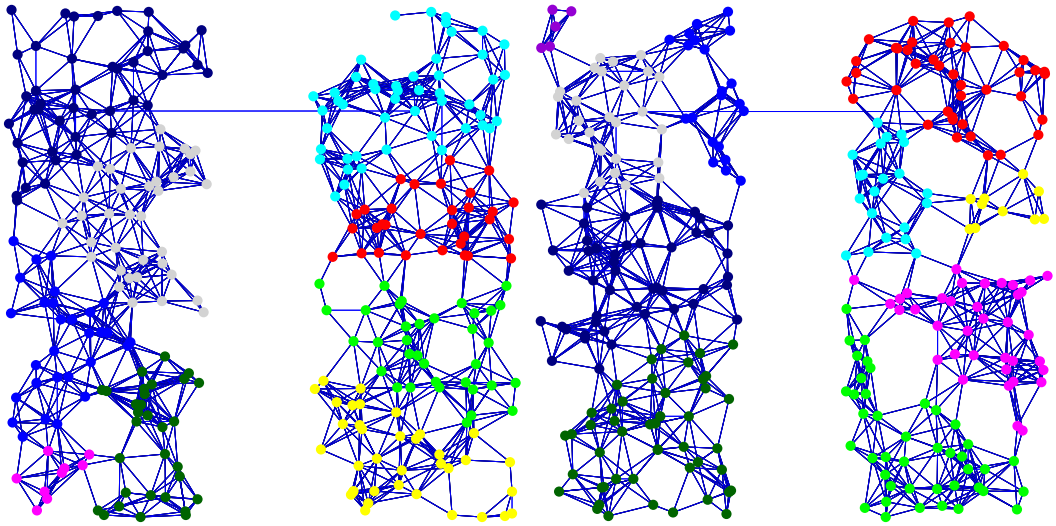
5. Each node sends a data packet periodically according to the received traffic pattern.

The traffic pattern consists of some traffic flow information: a source node, a destination node and a flow rate. In this chapter, flow rates are randomly selected from among $\frac{1}{10}$, $\frac{1}{20}$, $\frac{1}{30}$, $\frac{1}{40}$, $\frac{1}{50}$, $\frac{1}{60}$, $\frac{1}{70}$, $\frac{1}{80}$, $\frac{1}{90}$ and $\frac{1}{100}$. The number of flows is 0.2% of the number of all of the possible combinations of two nodes included in the VWSN topology. The pairs of source node and destination node are also selected randomly. The TTL of each data packet is set to 50.

From the simulations, we identified the reasons for failure to recover the data delivery ratio. These are listed in Table 3.3. Physical topology and the result of modular division have a strong influence on adaptivity within our proposed method and can lead to different reasons for failure to recover the data delivery ratio. Therefore, we use five physical topologies and perform 100 trials with each topology. In this chapter, we show the results of only two physical topologies; the results for the omitted topologies show the same characteristics. We call these physical topologies $T1$ and

Table 3.3: List of reasons for the failure to recover the data delivery ratio.

Abbreviation	Description of reasons for the recovery failure
$phyFrag$	The physical topology fragments into some subnetworks. This leads to unreachable nodes from a source node because no physical path exists between them.
$L0Frag$	A zeroth-tier virtual topology in the first-tier module fragments into some zeroth-tier virtual subnetworks. This leads to mutually unreachable pairs that belong to the same first-tier module, even when a physical path between them exists.
$L1Frag$	A first-tier virtual topology in the second-tier module fragments into some first-tier virtual subnetworks. This leads to unreachable pairs of a source first-tier module and a destination first-tier module that belong to the same second-tier module, even when a physical path between them exists.
$L2Frag$	A second-tier virtual topology in the third-tier module fragments into some second-tier virtual subnetworks. This leads to mutually unreachable pairs of second-tier modules that belong to the same third-tier module, even when a physical path between them exists.
$T0UF$	A node cannot find an alternative path to a node that belongs to the same first-tier module due to packet loss, when the zeroth-tier virtual topology in the first-tier module is not fragmented.
$T1UF$	A connected node that belongs to one first-tier module cannot find an alternative path to another first-tier module due to packet loss, when the first-tier virtual topology in the second-tier module is not fragmented.
$L0Loop$	A loop exists within the first-tier module due to inconsistent weights in the zeroth-tier routing table.
$L1Loop$	A loop exists within the second-tier module due to inconsistent weights in the first-tier routing table.
$LLLost$	A data packet cannot arrive at the end node of a virtual long link due to inconsistency in the long-link table.
TTL	A data packet expires, even when an alternative path is found.
$ConT^{L}Up$	A node cannot update its L-th-tier connection module table due to packet loss. Then, the node continues to send data packets to a module that is no longer a connected module.



(a) $T1$ (b) $T2$

Figure 3.7: Physical topology and the result of modular division by the Newman algorithm.

$T2$. Each physical topology with the result of modular division by the Newman algorithm is shown in Figure 3.7. Each color shows the group of each module.

3.7.3 Adaptivity of the VWSN Topology against Random Failure

In this section, we evaluate the adaptivity of the VWSN topology against random failure. The simulation time is 10,000 s; after 5000 s have elapsed, one node fails every 10 s until 30 nodes have failed. We use one hundred patterns of node failure, each corresponding to a trial. In each pattern, 30 nodes are randomly selected to fail, and the order of failure is random.

To compare the adaptivity of a physical topology with that of a VWSN topology, we try to evaluate the adaptivity of a physical topology in which we apply shortest-path routing to the topology. However, the simulation cannot be finished. Because all nodes have routing table entries to all other nodes, several nodes generate more than one hundred route request packets after failure of a single node. This leads to frequent packet collision and high packet loss. When a Hello packet is lost, neighboring nodes erroneously detect the failure of the sending node and generate many unnecessary route request packets. From this result, the division of a physical topology into small sub-topologies is an effective method for avoiding this type of problem.

Table 3.4 shows the rate of recovering the data delivery ratio in each combination of a physical topology and a method of constructing a VWSN topology. The recovery rate is the ratio between the number of trials in which all of the flows can reach the destination node and the total number of trials. In the simulation, phy indicates the physical topology. We use $R_{recovery}$ to denote the rate of recovering the data delivery ratio. Table 3.4 shows that $R_{recovery}$ for the bio-inspired

Table 3.4: Rate of recovering the data delivery ratio ($R_{recovery}$) against random failure.

	$\boldsymbol{phy}$	$\boldsymbol{R_{recovery}}$
BICM(hh,HH)	$T1$	0.80
BICM(ll,LL)	$T1$	0.81
Bio-inspired	$T1$	0.00
BICM(hh,HH)	$T2$	0.69
BICM(ll,LL)	$T2$	0.67
Bio-inspired	$T2$	0.00

topology is zero. There are some connected centroid nodes with a high degree in the virtual topology constructed by the bio-inspired method. The failure of a few of these is fatal because they are essential for connectivity. Note that the routing algorithm mentioned in Section 3.5 restricts the communication between modules because we keep nodes' overheads low by reducing their managing information on tables. A data packet must pass through the connected node of the module if the destination node of the data packet belongs to the different module. This means that, in the bio-inspired method, the virtual link connecting different modules is ignored when an endpoint node is neither a centroid node nor a peripheral node. Therefore, a VWSN topology constructed by the bio-inspired method seems to fragment easily into subnetworks because of the routing algorithm, even though it is highly robust on connectivity against random failure, as discussed in our previous work [41]. Moreover, these topologies generate many control packets as part of the route recovery mechanism in upper tiers. This results in congestion and the loss of control packets.

In the topologies created according to our proposal, the physical topology determines whether $R_{recovery}$ is high or low. As an example, when there are some sparse areas in the physical topology, it is difficult to find an alternative path by flooding because of collisions among control packets or isolation in the physical or virtual topology. When we compare values of $R_{recovery}$ among different

physical topologies, the differences are small between methods.

We investigate the cumulative distributions of $CP_{recovery}$ and $T_{recovery}$ against random failure; however, few differences can be seen among the strategies for $inter$ in each topology. Examples of the cumulative distributions of $CP_{recovery}$ and $T_{recovery}$ are shown in Figures 3.8 and 3.9, respectively. In Figure 3.8, $CP_{recovery}$ is mapped on the horizontal axis, and the cumulative number of trials in which the data delivery ratio recovers before $CP_{recovery}$ has elapsed is mapped on the vertical axis. Similarly, in Figure 3.9, the horizontal axis reflects $T_{recovery}$, and the vertical axis reflects the cumulative distribution of the number of trials in which the data delivery ratio recovers before $T_{recovery}$ has elapsed. Although the difference according to the method is small for the cumulative distribution of $CP_{recovery}$ or $T_{recovery}$ against random failure, when a VWSN is constructed by BICM(hh,HH), there are some trials whose $CP_{recovery}$ and $T_{recovery}$ are smaller (i.e., better) than those for topologies constructed by other methods. This is because when a node with a high degree is selected as a connected node in BICM(hh,HH), the endpoints of virtual links between modules tend to be concentrated to a small number of nodes. Therefore, the probability that a connected node fails in random failure is low, allowing the recovery of the zeroth-tier routing table unless a connected node fails.

The number of trials in which either the virtual or physical topology fragments is shown in Table 3.5, and the number of trials in which the data delivery ratio does not recover because of the inconsistency in tables is shown in Table 3.6. In this, phy identifies the physical topology. Blank entries in Table 3.5 reflect that the VWSN topology constructed by the bio-inspired method does not have tiers higher than the first tier. The values of $phyFrag$ are the same in each physical

Table 3.5: Number of trials in which the virtual or physical topology fragments when nodes are removed by random failure.

	$\boldsymbol{phy}$	$\boldsymbol{phyFrag}$	$\boldsymbol{L0Frag}$	$\boldsymbol{L1Frag}$	$\boldsymbol{L2Frag}$
BICM(hh,HH)	$T1$	19	17	1	0
BICM(ll,LL)	$T1$	19	17	0	0
Bio-inspired	$T1$	19	46	26	
BICM(hh,HH)	$T2$	6	18	5	0
BICM(ll,LL)	$T2$	6	18	5	0
Bio-inspired	$T2$	6	62	63	

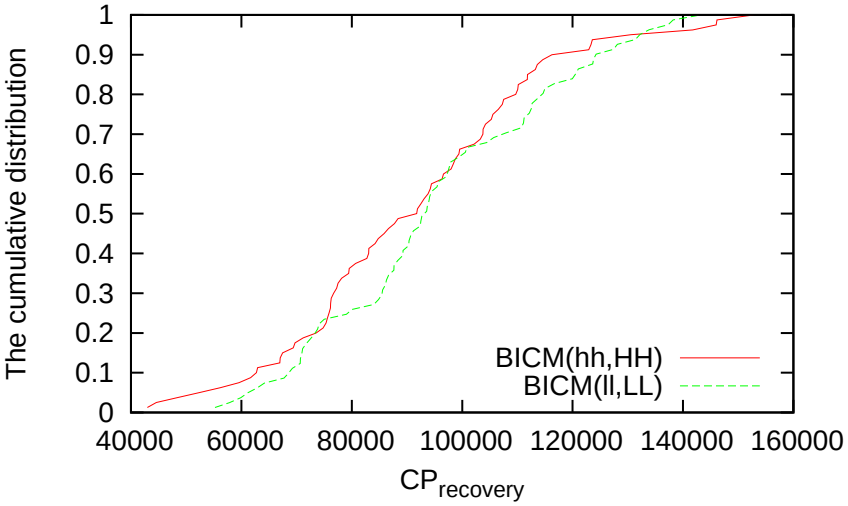


Figure 3.8: Cumulative distribution of the number of control packets ($CP_{recovery}$) when nodes are removed by random failure.

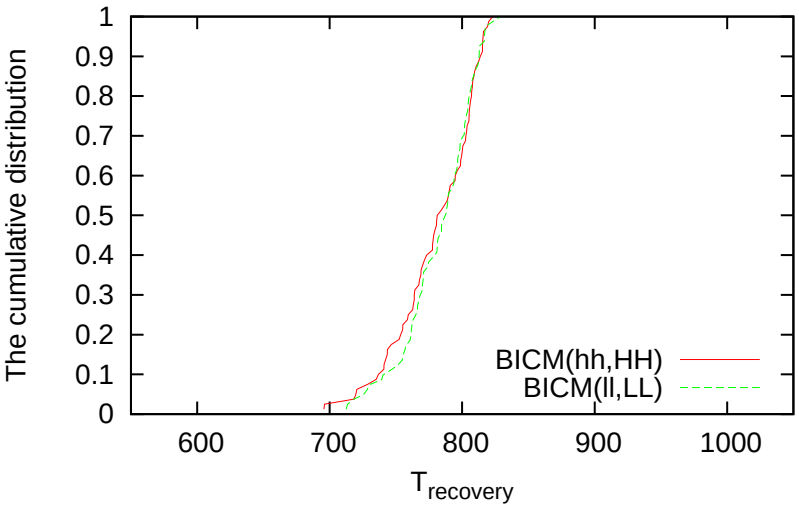


Figure 3.9: Cumulative distribution of the time needed for the data delivery ratio to recover to over 99.9% ($T_{recovery}$) when nodes are removed by random failure.

Table 3.6: Number of trials in which the data delivery ratio fails to recover due to the inconsistency in tables when nodes are removed by random failure.

	phy	$T0UF$	$T1UF$	$L0Loop$	$L1Loop$	$LLLost$	TTL	$ConT^{L}Up$
BICM(hh,HH)	$T1$	0	5	2	0	0	0	2
BICM(ll,LL)	$T1$	0	4	3	0	0	0	2
Bio-inspired	$T1$	11	99	2	0	0	0	2
BICM(hh,HH)	$T2$	0	14	0	0	0	0	1
BICM(ll,LL)	$T2$	3	13	1	0	0	0	2
Bio-inspired	$T2$	10	98	0	0	2	2	7

topology, because we use the same node failure patterns. The value of $L0Frag$ for BICM(hh,HH) and BICM(ll,LL) is the same because the results of modular division by the Newman algorithm are almost identical between the cases. Whether the value of $L1Frag$ is high or low depends on the physical topology. For topologies constructed according to our proposed method, it is easy to fragment the first-tier virtual topology into physical topology $T2$. There are two main reasons for this. One is that there is a first-tier module with a small number of nodes, and the other is that there is a first-tier module whose degree in the first tier is one. The point in common between these is that some first-tier modules contain few connected nodes. Therefore, the failure of a few connected nodes in a first-tier module that either has few nodes or has degree one in the first tier results in the fragmentation of the module. When we use the bio-inspired method, few connected nodes exist in their respective first-tier modules, and each module is small. Therefore, fragmentation of the topology is more likely than with our proposed method.

In Table 3.6, $T1UF$ has the highest value for each method. This is because it is comparatively difficult to find an alternative route in the upper tier. Particularly when using the bio-inspired method, $T1UF$ is nearly the same as the number of trials. This near parity results from the congestion and loss of control packets around the connected nodes, which attract a large number of links.

From the above, the adaptivity of VWSN topologies constructed by our proposal is not markedly different. However, in a few trials, $T_{recovery}$ is notably smaller in the topology constructed by BICM(hh,HH). This suggests that the adaptivity to random failure of a VWSN topology constructed by BICM(hh,HH) is comparatively high.

3.7.4 Adaptivity of the VWSN Topology against Targeted Attacks

In this section, we evaluate the adaptivity of the VWSN topology against targeted attacks. The simulation time is 10,000 s; after 5000 s have elapsed, one node fails for every 10 s until 30 nodes have failed. Nodes fail in descending order of initial degree in the VWSN topology, without adjusting degrees after each failure and choosing arbitrarily among nodes of equal degree.

The values of $R_{recovery}$, average $T_{recovery}$ and average $CP_{recovery}$ for each method are summarized in Table 3.7. The number of trials in which the virtual or physical topology becomes fragmented is shown in Table 3.8, and the number of trials in which the data delivery ratio fails to recover due to the inconsistency in the tables is shown in Table 3.9. As elsewhere, phy indicates the physical topology. The blanks in Table 3.7 indicate that the value could not be calculated because the data delivery ratio did not recover.

Table 3.7: The rate of recovering the data delivery ratio ($R_{recovery}$), the average time needed for data delivery ratio to recover to over 99.9% ($T_{recovery}$) and the average number of control packets ($CP_{recovery}$) when nodes are removed by targeted attack.

	phy	$R_{recovery}$	Average $T_{recovery}$	Average $CP_{recovery}$
BICM(hh,HH)	$T1$	0.16	793.43	16.66×10^4
BICM(ll,LL)	$T1$	0.00		
Bio-inspired	$T1$	0.00		
BICM(hh,HH)	$T2$	0.07	791.87	24.13×10^4
BICM(ll,LL)	$T2$	0.83	797.07	18.17×10^4
Bio-inspired	$T2$	0.00		

Table 3.8: Number of trials in which the virtual or physical topology becomes fragmented when nodes are removed by targeted attack.

	phy	$phyFrag$	$L0Frag$	$L1Frag$	$L2Frag$
BICM(hh,HH)	$T1$	0	35	48	26
BICM(ll,LL)	$T1$	0	100	0	0
Bio-inspired	$T1$	0	88	86	
BICM(hh,HH)	$T2$	1	7	61	9
BICM(ll,LL)	$T2$	0	8	0	0
Bio-inspired	$T2$	0	100	99	

Table 3.9: Number of trials in which the data delivery ratio fails to recover due to the inconsistency in tables when nodes are removed by targeted attack.

	phy	$T0UF$	$T1UF$	$L0Loop$	$L1Loop$	$LLLost$	TTL	$ConT^{L}Up$
BICM(hh,HH)	$T1$	0	49	1	1	0	2	2
BICM(ll,LL)	$T1$	0	2	4	0	1	0	1
Bio-inspired	$T1$	7	100	2	0	2	0	11
BICM(hh,HH)	$T2$	3	57	6	0	0	1	10
BICM(ll,LL)	$T2$	7	1	2	0	0	1	1
Bio-inspired	$T2$	8	100	5	0	1	1	8

In physical topology $T2$, $R_{recovery}$ is quite low when using BICM(hh,HH), but $R_{recovery}$ remains high when using BICM(ll,LL). When using BICM(hh,HH), it is easy to divide a module, and so, the route discovery process must be carried out frequently because high-degree nodes are selected to be connected nodes. This means that many connected nodes fail when nodes are removed by targeted attack. However, the value of $R_{recovery}$ is zero in BICM(ll,LL) for $T1$. This is because a zeroth-tier topology in a first-tier module becomes fragmented in all trials, as shown in Table 3.8. In $T1$, there is a first-tier module whose central area is densely connected. Because the degree of a node in this area is high, this first-tier module becomes fragmented when all of the nodes in the area fail.

From the above, although adaptivity against targeted attack is strongly dependent on physical topology and the method of modular division, the adaptivity of the VWSN constructed by BICM(ll,LL) is the highest in many cases.

Considering all of the results, the physical deployment of sensor nodes and the modular division algorithm is quite important for keeping services running on a VWSN topology. Because sparse areas where link density is low could fragment easily, infrastructure providers should deploy redundant nodes in order to provide stable services. Nodes in dense areas where link density is high can consume more energy than nodes in other areas because they will forward or receive more packets. When a first-tier module has both dense and sparse areas, the energy depletion of nodes in the dense area can result in the fragmentation of the module. To prevent this, infrastructure providers should supply energy to nodes or deploy nodes with high-capacity batteries in areas where many nodes are

to be deployed. Alternatively, VWSN providers should construct a first-tier module in which the link density is homogeneous.

3.7.5 Discussion of Memory Utilization

In this section, we estimate the amount of memory needed to store the tables used for the routing algorithm shown in this chapter. For the L-th-tier routing table, the number of entries that each node needs to hold is the sum of the number of L-th-tier modules that belong to the $(L+1)$-th-tier module of that node (excluding the L-th-tier module of the node) and the number of neighboring L-th-tier modules that belong to a different $(L+1)$-th-tier module. Therefore, the number of entries $Z_{RT}^{L}(n)$ of the L-th-tier routing table when node n belongs to an L-th-tier module M_a^L and an $(L+1)$-th-tier module M_x^{L+1} is defined as follows:

$$Z_{RT}^{L}(n) = |Comp^{L}(M_x^{L+1})| - 1 + |\mathcal{N}^{L}(M_a^{L}) \setminus Comp^{L}(M_x^{L+1})| \tag{3.6}$$

Here, $n \in M_a^L$ and $n \in M_x^{L+1}$. $Comp^{L}(M_x^{L+1})$ is the set of L-th-tier modules that belong to the $(L+1)$-th-tier module M_x^{L+1}, and $|Comp^{L}(M_x^{L+1})|$ is the cardinality of $Comp^{L}(M_x^{L+1})$. The second term means that M_a^L is not included as the destination module in the L-th-tier routing table. $\mathcal{N}^{L}(M_a^{L})$ is the set of L-th-tier modules that neighbor M_a^L, and $\mathcal{N}^{L}(M_a^{L}) \setminus Comp^{L}(M_x^{L+1})$ is the relative complement of $Comp^{L}(M_x^{L+1})$ in $\mathcal{N}^{L}(M_a^{L})$. Then, $|\mathcal{N}^{L}(M_a^{L}) \setminus Comp^{L}(M_x^{L+1})|$ is the number of L-th-tier modules that are connected to M_a^L and belong to an $(L+1)$-th-tier module other than M_x^{L+1}. From the above, the total number of entries $Z_{RT}(n)$ of the routing tables held by node n is

$$Z_{RT}(n) = \sum_{i} Z_{RT}^{i}(n), \tag{3.7}$$

where i is the identity of each tier composing the VWSN topology.

Because the number of zeroth-tier modules, which are nodes belonging to the same first-tier module, is the largest of all of the tiers, the size of the zeroth-tier routing table is dominant in many cases.

For the L-th-tier connection table, the number of entries that each node needs to hold is the

number of $(L-1)$-th-tier virtual links that connect an $(L-1)$-th-tier module belonging to the L-th-tier module of the node with another $(L-1)$-th-tier module belonging to a neighboring L-th-tier module. In our proposal, the number of $(L-1)$-th-tier virtual links added by mapping the L-th-tier virtual link depends on the number of $(L-1)$-th-tier virtual links embedded in the L-th-tier VWSN topology. Therefore, the number of entries $Z_{CT}^{L}(n)$ of an L-th-tier connection table for node n that belongs to M_a^L is as follows:

$$Z_{CT}^{L}(n) = \sum_{M_y^L \in \mathcal{N}^L(M_a^L)} \left\lceil C_{\text{inter}}^{L} \left(\left| E^{M_a^L} \right| + \left| E^{M_y^L} \right| \right) \right\rceil \tag{3.8}$$

Here, $n \in M_a^L$. Therefore, the total number of entries $Z_{CT}(n)$ of the connection tables held by n is

$$Z_{CT}(n) = \sum_{i} Z_{CT}^{i}(n), \tag{3.9}$$

where i is the identity of each tier composing the VWSN topology.

Although the number of zeroth-tier virtual links is the largest among all of the tiers, we can tune the parameter C_{inter}^{L} separately for each tier. In this chapter, because we set C_{inter}^{L} to a small value, the size of the connection tables is smaller than that of the routing tables.

For the long-link table, the number of entries that each node needs to hold, denoted by $Z_{LT}(n)$, is the number of virtual links to which it is assigned as a relay node or an end node. Because the number of entries of a long-link table depends entirely on the specific node, we cannot easily estimate the size of the long-link table. Some nodes will have an empty long-link table; others will have a large number of entries in the long-link table. In our evaluation environment, the two nodes connected by the wired link and the nodes around those end nodes have a large number of entries in their long-link tables, because all traffic between the two sensor networks must go through the wired link. However, the number of entries of the long-link tables is much less than that of the routing tables because of the large number of zeroth-tier modules. Because the sizes of the long-link tables depend on the method of modular division, further investigation into the effect of the choice of method of modular division will be needed.

Then, we derive the expectation of the total number of entries of each L-th-tier table, denoted

by $\langle Z_{RT}^L \rangle$, $\langle Z_{CT}^L \rangle$ and $\langle Z_{LT}^L \rangle$, respectively. Let us define the expectation of each variable as follows. $\langle |Comp^L(M^{L+1})| \rangle$, $\langle \left| E^{M^L} \right| \rangle$ and $\langle k_{M^L} \rangle$ denote the expectation of the number of L-th-tier modules belonging to an $(L+1)$-th-tier module, the expectation of the number of $(L-1)$-th-tier virtual links in an L-th-tier module and the expectation of the degree of an L-th-tier module, respectively.

We calculates the expectation of the number of L-th-tier modules that are connected to an L-th-tier module and belong to an $(L+1)$-th-tier module other than the $(L+1)$-th-tier module that the L-th-tier module belongs to, denoted by $\langle |\mathcal{N}^L(M_a^L) \setminus Comp^L(M_x^{L+1})| \rangle$. The expectation of the total number of L-th-tier virtual links that connected L-th-tier modules belonging to an $(L+1)$-th-tier module and L-th-tier modules belonging to other $(L+1)$-th-tier modules equals $\langle Z_{CT}^{L+1} \rangle$. Then, the probability that the number of L-th-tier virtual links connecting an L-th-tier module and other L-th-tier modules belonging to other $(L+1)$-th-tier modules is χ, denoted by θ, is as follows:

$$\theta = \binom{\langle Z_{CT}^{L+1} \rangle}{\chi} \left(\frac{1}{\langle |Comp^L(M^{L+1})| \rangle} \right)^{\chi} \left(1 - \frac{1}{\langle |Comp^L(M^{L+1})| \rangle} \right)^{\langle Z_{CT}^{L+1} \rangle - \chi} \tag{3.10}$$

Because of the binomial distribution, the following satisfies:

$$\langle |\mathcal{N}^L(M_a^L) \setminus Comp^L(M_x^{L+1})| \rangle = \frac{\langle Z_{CT}^{L+1} \rangle}{\langle |Comp^L(M^{L+1})| \rangle} \tag{3.11}$$

Then, $\langle Z_{RT}^L \rangle$ is calculated as follows:

$$\langle Z_{RT}^L \rangle = \langle |Comp^L(M^{L+1})| \rangle - 1 + \frac{\langle Z_{CT}^{L+1} \rangle}{\langle |Comp^L(M^{L+1})| \rangle} \tag{3.12}$$

From Equation (3.8), $\langle Z_{CT}^L \rangle$ is calculated as follows:

$$\langle Z_{CT}^L \rangle = \left\lceil 2C_{\text{inter}}^L \langle k_{M^L} \rangle \langle \left| E^{M^L} \right| \rangle \right\rceil \tag{3.13}$$

As mentioned above, we cannot easily estimate the size of the long-link table. Therefore, we use $\langle Z_{LT}^L \rangle$ as a parameter in the following discussion.

Then, we discuss how much memory size is required for the tables. Let us denote b_r^L, b_c^L and b_l as the required size of memory of an entry of L-th-tier routing table, L-th-tier connection table

and long-link table in bytes, respectively. Then, the required total memory size for tables of node n, denoted by $B(n)$, is

$$B(n) = \sum_i \left(b_r^i Z_{RT}^i(n) + b_c^i Z_{CT}^i(n) \right) + b_l Z_{LT}(n). \tag{3.14}$$

The entries in the routing table in the zeroth tier consist of a destination node, a forwarding node for the destination node and path weight. The entries in the routing table for the higher tier consist of an identifier of a destination module, a forwarding module for the destination module and path weight. Because of the difference of the scale of the number of nodes/modules in each tier, the memory size for the identifier of modules can be smaller in a higher tier. Therefore, we assume that the memory size of the identifier for nodes and modules are two bytes and one byte, respectively, here. We treat the path weight as the float type variable (four bytes). Then, $b_r^0 = 8$ and $b_r^L = 6$ where $L > 0$.

When node n belongs to the L-th-tier virtual module M_A^L, it holds an L-th-tier connection table whose entries consist of L-th-tier neighboring module identifiers (M_B^L), the module identifier of $(L-1)$-th-tier connected modules (M_X^{L-1}) belonging to M_A^L and the module identifiers of $(L-1)$-th-tier connected modules (M_Y^{L-1}) belonging to M_B^L. Because a zeroth-tier module denotes a node, $b_c^1 = 6$ and $b_c^L = 4$ where $L > 1$.

The entries of a long-link table consist of the end nodes of a virtual long-distance link, the next and previous forwarding nodes and a hop count from the source node of the virtual long-distance link. Because we set TTL to 50 in our evaluation, one byte is enough for a hop count from the source node of the virtual long-distance link. Then, $b_l = 9$.

From the above assumptions, Equation (3.14) can be rewritten as

$$B(n) = 8Z_{RT}^0(n) + \sum_{i>0} \left(6Z_{RT}^i(n) \right) + 6Z_{CT}^1(n) + \sum_{i>1} \left(4Z_{CT}^i(n) \right) + 9Z_{LT}(n). \tag{3.15}$$

Now, we assume a two-tiered structure for one application and a situation in which $\langle |Comp^0(M^1)| \rangle = 100$, $\langle |Comp^0(M^2)| \rangle = 10$, $\langle \left| E^{M^1} \right| \rangle = 2000$, $\langle k_{M^1} \rangle = 5$ and $\langle Z_{LT}(n) \rangle = 10$. Because the expected number of nodes in a first tier module is 100 and the expected number of total first-tier modules is

10, the total number of nodes in the VWSN network is 1000. From Equations (3.12) and (3.13), the expectation of the required total memory size for tables of a node is 2152 bytes.

Next, we assume a three-tiered structure for one application and a situation in which $\langle |Comp^0(M^1)| \rangle = 100$, $\langle |Comp^1(M^2)| \rangle = 10$, $\langle |Comp^2(M^3)| \rangle = 3$, $\langle \left| E^{M^1} \right| \rangle = 2000$, $\langle \left| E^{M^2} \right| \rangle = 20$, $\langle k_{M^1} \rangle = 5$, $\langle k_{M^2} \rangle = 2$ and $\langle Z_{LT}(n) \rangle = 20$. In this case, the total number of nodes in the VWSN network is 3000. Then, the expectation of the required total memory size for tables of a node is 2290.8 bytes.

Then, we assume a four-tiered structure for one application and a situation in which $\langle |Comp^0(M^1)| \rangle = 100$, $\langle |Comp^1(M^2)| \rangle = 10$, $\langle |Comp^2(M^3)| \rangle = 3$, $\langle |Comp^3(M^4)| \rangle = 3$, $\langle \left| E^{M^1} \right| \rangle = 2000$, $\langle \left| E^{M^2} \right| \rangle = 20$, $\langle \left| E^{M^3} \right| \rangle = 3$, $\langle k_{M^1} \rangle = 5$, $\langle k_{M^2} \rangle = 2$, $\langle k_{M^3} \rangle = 3$ and $\langle Z_{LT}(n) \rangle = 30$. In this case, the total number of nodes in the VWSN network is 9000. Then, the expectation of the required total memory size for the tables of a node is 2404.8 byte. Therefore, there is a case that 3 KB of memory on average per node is enough for the routing algorithm shown in this chapter even when the four-tiered VWSN is provided.

Note that tables for the routing can be reused by multiple applications. When a user wants to run an application over the integrated VWSN, which comprises multiple VWSNs that may have been deployed for other applications, the same tables for routing in the lower tiers can be used in the integrated VWSN. Additional entries in the tables for routing are needed for the highest tier only. Therefore, the modular structure contributes to memory efficiency in such situations. Moreover, in our design, multiple routing policies can be set in the module for each services in case of avoiding potential problems, such as overload on certain physical links or nodes.

3.7.6 Discussion of Approaching Our Considered World from the Current Real World

In this chapter, we show an overall architecture that is suitable for constructing and running VWSN services within a VWSN topology for the IoT environment. However, it is often considered that nodes in the WSNs are involved in severe restriction on their processing, energy, memory and storage. In the virtualization scenario, this restriction is more critical because multiple applications, such as in-network processing defined by users or the manager of concurrency, run on the same

entity.

The things to process are concurrency management, protocol translation, sensing, actuation, packet processing, signal processing, timer management, routing, route recovery, neighbor nodes' management and table management. Because they can add a big overhead to nodes, how to manage device resources is a crucial viewpoint of network design. In completely centralized management systems, an overhead for collecting information of nodes explosively grows as the number of nodes in the system increases. However, on the contrary, in decentralized systems, more powerful resources are required for any node in the network.

Khan et al. mentioned that research of the virtualization of WSNs is getting more pertinent because WSNs' nodes are becoming more powerful [56]. It is expected that this trend will continue, and WSNs' nodes will get more powerful resources in the future. In our architecture, therefore, sensor nodes process their tasks in a decentralized manner after the VWSN provider deploys the constructed VWSN.

Moreover, some techniques with small overheads can be applied to our architecture. For example, only high-spec connected nodes hold the entire L-tier routing tables and are responsible for routing between higher tier modules, while a low-spec node holds only zeroth-tier table and sends its sensing data to only the nearest connected node. Any routing algorithm will do in the zeroth-tier, not only any-to-any routing, but also converge routing to a high-spec node. Many efficient converge routing algorithms for WSNs have been proposed [73]. Here, a method of resource management or an energy-efficient solution is out of the scope of this chapter and relies on other research.

Another important aspect is Internet compatibility. It is natural to consider that the VWSN providers or users access the virtualized resources through the Internet. Therefore, we need to address compatibility or connectivity between WSNs and the Internet. As mentioned by many researchers [12], the gateway-based strategy can be one solution to this problem. Our architecture can also gain the compatibility to the Internet by using gateway-based solutions. In our architecture, connected nodes can be seen as gateways between modules or networks. As mentioned in Section 3.1, we consider that users can select or configure protocols that they use in their applications. Standardized protocols, such as 6LoWPAN, can be also included. Moreover, this idea can be applied to each tier in a VWSN network deployed for an application. As mentioned above, because

there are some low-spec nodes in WSNs, an energy-efficient and low overhead routing algorithm can be the first choice in zeroth-tier networks. Then, each connected node, behaving as a gateway, aggregates data packets whose destination is out of the module, encapsulates them for routing in higher tier networks and translates protocols as necessary. In this scenario, although the amount of energy consumption is heterogeneous, the heterogeneity of the nodes' spec is more general in the future IoT environment because of the existence of multiple vendors and providers. How to manage such heterogeneity is out of our scope, but should be future work.

3.8 Summary

In a scenario of the virtualization of WSNs, physical networks can undergo dynamic change, such as the addition or removal of nodes or links or resource assignment to fulfill new user requests. Therefore, reliability is important even when such environmental changes occur. To tackle these problems, we show an overall architecture of constructing and running VWSN services with considering the environmental changes. We define that reliability consists of robustness and adaptivity. In our previous work, we proposed a method for constructing a robust VWSN topology against node failure. In this chapter, we conduct a simulation of the practical situation to evaluate the adaptivity of our proposed VWSN topology in consideration of an actual environment.

The results of the simulation experiments showed that the adaptivity of the VWSN topology constructed by BICM(ll,LL) was the highest against target attack, which is consistent with the robustness results. Thus, we do not only provide redundancy of connectivity but also provide a structure in which it is easy to find an alternative route between modules by avoiding congestion on a small number of boundary nodes. This ability of adaptive configuration in VWSN can be gained by connecting lower-degree modules. Therefore, we mention that it is important to focus on degrees when we consider the communication between virtual modules. However, when there is a first-tier module whose size is small or a first-tier module whose central area is densely connected, the zeroth-tier virtual topology in the first-tier module becomes fragmented quite easily. To address this problem, it is necessary to more deeply consider the method of modular division and the modular structure. In this chapter, we assume that all physical links have the same quality. Therefore, a

means of constructing a robust and adaptive VWSN topology with taking the difference of the link quality into consideration should be discussed in future work.

We also discussed the memory needed for the tables used in our proposed routing. In our evaluation environment, the size of a zeroth-tier routing table is the most dominant. However, the number of entries of each table depends on the method of modular division, the number of connected modules and nodes and the number of virtual long-distance links. Therefore, further investigation into the relation between the chosen method of modular division and memory efficiency will be needed.

In this chapter, we analyzed the properties of a virtual topology composed of only sensor nodes. Therefore, considering the following is necessary. First, the method of realizing a virtual link in a physical network should be investigated; packets forwarded along a virtual long-distance link should be conveyed with only a short delay. As candidate methods for this, we intend to consider creating directional beams, increasing omnidirectional transmission range and multi-hop forwarding with variable priority. Second, because there may be multiple demands for constructing VWSNs that compete for resources, such as energy, memory and bandwidth, it is necessary to consider a method that can construct resource-efficient VWSN topologies. Third, we hope to create a protocol for evolving the VWSN topology in response to environmental changes, such as changes in traffic patterns. Due to the modular structure, a small adjustment of a few virtual links between modules should be sufficient to achieve that.

Appendices

3.A Packet Format

In this section, we explain the packet format used to realize the routing mentioned in Section 3.5.1. The data packet needs to include an application identifier, a source node address, an identifier of the i-th-tier module to which the source node belongs ($1 \leq i \leq L$), a destination node address, an identifier of the i-th-tier module to which the destination node belongs and the sender and receiver node addresses of the data packet. These pieces of information are included in the packet by encapsulating a packet recursively and adding the identifiers for the source and destination module in each tier. The packet format is shown in Figure A1. The source node encapsulates the data packet. When the L-th-tier packet reaches the destination module in the L-th tier, it is unpacked, and the $(L-1)$-th-tier packet is then routed.

The field "long-distance link mode" is a flag that shows whether the data packet exists on the path assigned to a virtual long-distance link and is routed to the end node of the virtual long-distance link. The field "inter module mode" is a flag that shows whether the data packet exists on the path assigned to a virtual link connecting modules in the first tier or higher. The field "hop count on long-distance link" is the hop count to the source node of a virtual long-distance link, and the field containing the "path on long-distance link" is a list of node addresses for forwarding this packet along the virtual long-distance link. These fields are used for the maintenance of long-link tables.

If the data packet reaches the source node a of a virtual long-distance link and node a decides to forward it to destination node b of the virtual long-distance link, then node a encapsulates the packet to indicate that the packet is in long-distance link mode. Then, node a queries its long-link table and

determines the next-hop node c for delivering the packet to node b along the virtual long-distance link. The packet format for this situation when the data packet exists on a virtual long-distance link is shown in Figure A2.

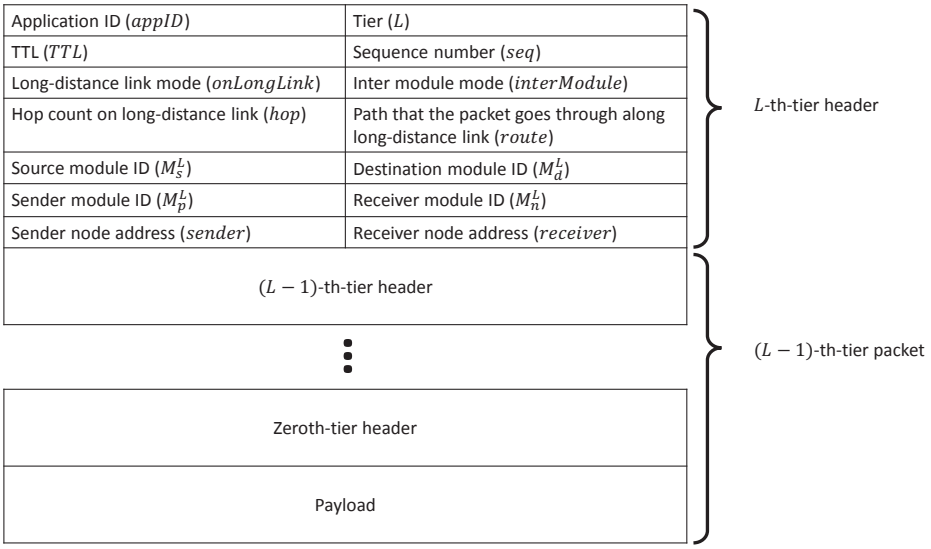


Figure A1: Format of the L-th-tier header prepended to $(L-1)$-th-tier packets.

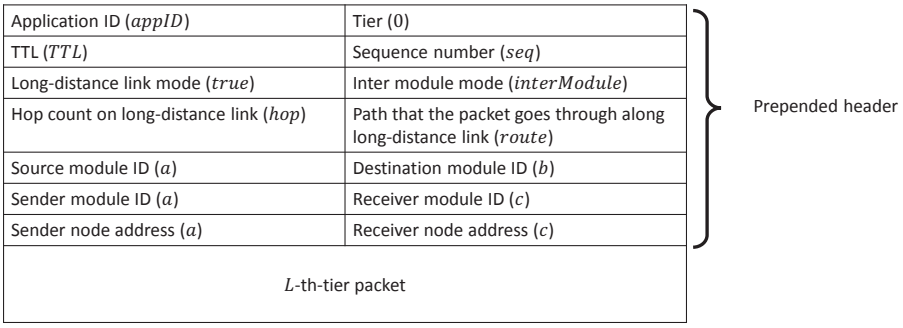


Figure A2: Packet format when the data packet exists on a virtual long-distance link.

3.B Definition of Weights of Virtual Links and Virtual Modules

In this section, we define the weights of virtual links and virtual modules used in selecting a forwarding module by the method described in Section 3.5.1. In this chapter, the weights of virtual

links and virtual modules are calculated from the hop counts of the physical paths.

First, we define the weight of an L-th-tier virtual link. Because a virtual module in the zeroth tier is identically a physical node, we define the weight of a zeroth-tier virtual link as the shortest hop count between its end nodes. The definition of the weight of an L-th-tier virtual link whose L-th-tier endpoint modules are M_A^L and M_B^L is

$$W_l^L(M_A^L, M_B^L) = \frac{1}{|P^{L-1}(M_A^L, M_B^L)|} \sum_{(x,y)\in P^{L-1}(M_A^L, M_B^L)} W_l^{L-1}(x, y), \tag{B1}$$

where $P^{L-1}(M_A^L, M_B^L)$ is the set of $(L-1)$-th-tier virtual links to which the L-th-tier virtual link (M_A^L, M_B^L) is assigned; $|P^{L-1}(M_A^L, M_B^L)|$ is the cardinality of $P^{L-1}(M_A^L, M_B^L)$; and $W_l^{L-1}(x, y)$ is the weight of the $(L-1)$-th-tier virtual link between module x and module y. The weight of the L-th-tier virtual link between module M_A^L and module M_B^L is the mean of the weights of the $(L-1)$-th-tier virtual links to which it is assigned.

Then, we define the weight of an L-th-tier virtual module as the cost of going through the module. We assume that the weight of a virtual module in the zeroth tier is zero. We define the weight of an L-th-tier virtual module as the mean of the hop count from one neighboring L-th-tier module to another neighboring L-th-tier module. This means that the weight of a virtual module changes according to the source and destination of a flow. Therefore, when a flow comes from an L-th-tier virtual module M_B^L and goes to M_C^L, we define the weight of an L-th-tier virtual module M_A^L as

$$W_m^L(M_A^L, M_B^L, M_C^L) = \frac{1}{|Con(M_B^L, M_A^L)| \cdot |Con(M_C^L, M_A^L)|} \sum_{\substack{x\in Con(M_B^L, M_A^L),\\ y\in Con(M_C^L, M_A^L)}} PW(M_A^L, x, y), \tag{B2}$$

where $Con(M_B^L, M_A^L)$ is the set of $(L-1)$-th-tier connected modules belonging to M_B^L and connecting to an $(L-1)$-th-tier connected module of M_A^L, and $|Con(M_B^L, M_A^L)|$ is the cardinality of $Con(M_B^L, M_A^L)$. Then, $PW(M_A^L, x, y)$ is the weight of the virtual module M_A^L when a flow comes from an $(L-1)$-th-tier module x and goes to an $(L-1)$-th-tier module y. We define $PW(M_A^L, x, y)$

as

$$PW(M_A^L, x, y) = \min_{p \in Path^{L-1}(M_A^L, x, y)} \Bigg(W_m^{L-1}(M_{r_1}^{L-1}, x, M_{r_2}^{L-1}) + \sum_{i=2}^{\kappa-1} \big(W_l^{L-1}(M_{r_{i-1}}^{L-1}, M_{r_i}^{L-1}) + W_m^{L-1}(M_{r_i}^{L-1}, M_{r_{i-1}}^{L-1}, M_{r_{i+1}}^{L-1}) \big) + W_l^{L-1}(M_{r_{\kappa-1}}^{L-1}, M_{r_\kappa}^{L-1}) + W_m^{L-1}(M_{r_\kappa}^{L-1}, M_{r_{\kappa-1}}^{L-1}, y) \Bigg), \tag{B3}$$

where $Path^{L-1}(M_A^L, x, y)$ is the set of $(L-1)$-th-tier paths between x and y. Each element p is represented as $(M_{r_1}^{L-1}, \cdots, M_{r_\kappa}^{L-1})$, where elements of $M_{r_i}^{L-1} (1 \leq i \leq \kappa)$ belong to M_A^L.

Chapter 4

Percolation Analysis for Constructing a Robust Modular Topology based on a Binary-dynamics Model

4.1 Introduction

In the context of Internet of Things, various devices, such as sensor nodes or actuator nodes, are deployed all over the world and each subset of them compose a local network. When we consider user's various service demands, these networks are required to cooperate with each other. Therefore, mechanisms for sharing networks as infrastructure are quite important. As a crucial technique for sharing substrates of networks, virtualization of wireless sensor networks (WSNs) has been attracting a great deal of attention [34]. One way of virtualization of WSN is to construct a logically connected overlay network for each application. In a virtualization scenario, multiple sensors in multiple WSNs can be used as shared infrastructure, with some sensors integrated for running each application. The virtualization of WSNs improves manageability and flexibility.

When we integrate local networks, modular structure emerges. A module consists of a group of nodes connected densely by a large number of intra-module links and modules are connected sparsely by a few number of inter-module links. Such modular structure can be seen in many real

networks such as Internet, social networks or biological networks. An artificial network integrated by interdependent systems is highly vulnerable to targeted attacks or cascading failures and results in fragmentation [74]. In sensor networks, the robustness of the connectivity is a crucial issue because the robust structure of the networks leads to low overhead of maintaining services running over them stable. The study of the way of connecting sensor networks to earn the robustness thus leads to improve the cooperation of deployed sensor devices. Therefore, it is important to investigate effect of modular connection pattern on efficiency of a network, especially robust connectivity.

In our previous work, we proposed a brain-inspired method for constructing a robust and adaptive virtual wireless sensor network (VWSN) topology [41]. We showed that the method of constructing links between modules has crucial effect on robustness and adaptivity of the resulting VWSN topology [41]. However, the best way of constructing a robust and adaptive VWSN topology, and particularly of constructing links between modules, is still unclear. Toward clarifying this, we use an analytical method to study robustness of a topology according to the method of constructing links between modules. Note that we regard a network topology as an undirected graph in analytical theory.

In this chapter, we use an analytical approach and show the way of connecting modules so that a constructed network has the most robust connectivity. We propose a method to investigate percolation dynamics on a modular network, especially graph ensembles after addition of inter-module links. In consideration of addition of inter-module links, we add a new tool to a binary-dynamics model [49] which is an analytical method for estimating robustness of modular networks.

Our main contribution is that our analytical method considers the link addition and can be applied to make a policy for embedding a new link. Existing studies can be applied only for estimating robustness of given graph ensembles. However, our proposal enables to investigate percolation behavior according to different embedding patterns of inter-module links when a probability distribution of intra-module graph ensembles and that of inter-module link ensembles are given independently. Note that rewiring strategy in which the probability distribution does not change can make the problem for the analytical theory simple. However, link additions are more general than rewiring in the actual environment. Also, our virtual topology construction method proposed in our previous work [41] is composed of constructing intra-module topology and adding inter-module

links to connect them. Therefore, in this paper, we focus on the link additions for improving our method proposed in our previous work [41].

Through simulation evaluations, our analytical results are in good agreement with numerical simulations in a configuration model network. Additionally, we show that it is hard to apply our proposal to the graph in which the number of nodes is small because the target of our approach is average properties of random graph ensembles. For the similar reason, we show that the result of analysis cannot completely capture the result of a percolation process on a graph having special structural properties, such as ring-shaped structure.

4.2 Related Work

Many researchers have studied percolation processes on various types of graphs by using a generating function approach. In this type of approach, the expected size of the giant component of a random graph ensemble can be derived from a probability distribution, as given by a degree distribution or a distribution of types of links. We can then estimate robustness of a graph by evaluating percolation transitions of the size of the giant component. However, prior studies have focused on estimating robustness of given graph ensembles and not considered changes to graphs, such as link additions.

A generating function approach has been proposed for estimating robust connectivity of random graph ensembles. In this method, the targeted ensemble of random graphs is defined by a generating function $\mathcal{G}(x)$, which represents a probability distribution, such as the degree distribution, by using an auxiliary variable x. A generating function for the probability distribution of component sizes, denoted by $\mathcal{H}(x)$, can then be calculated from $\mathcal{G}(x)$. When a giant component exists, we can calculate the size of the giant component by calculating the ratio of nodes that do not belong to the giant component, from $\mathcal{H}(x)$. In this research area, many researchers have studied percolation processes on various types of graphs, such as random graphs [45], networks of networks [46], multiplex networks [47] and interdependent networks [48]. However, the generating function approach is complex because many auxiliary variables and generating functions are necessary, differing according to the complexity of the targeted graphs.

Another important analytical method is a binary-dynamics model for evaluating percolation and other dynamic processes [49]. This method is relatively simple. In this method, the probability distribution of links is used to obtain the percolation behavior of the network. The probability distribution of links represents modular structure, degree–degree correlation within modules, and degree–degree correlation between modules. The type of the percolation model can be configured by changing the definition of a response function. The detail of the binary-dynamics model is shown in the next section. The binary-dynamics model can be applied to broad classes of graphs by configuring the probability distribution according to the node types or link types.

Almost all existing methods, however, focus on evaluating percolation processes on graph ensembles where a probability distribution is given, and the problem of how to embed links for constructing a robust topology is not examined. Therefore, we propose an analytical method that takes into account changes in the probability distribution due to addition of inter-module links.

4.3 Method

4.3.1 Binary-dynamics model

Before we explain our proposal, we explain the binary-dynamics model in detail. For convenience of explanation, we denote the type of a degree-k node belonging to module i by (i, k) and the type of a link that connects (i, k) node with (i', k') node by $\{(i, k), (i', k')\}$. The probability distribution of links is then defined by tensor $[P_{k,k'}^{i,i'}]$ in which each element represents the probability that a randomly chosen link is an $\{(i, k), (i', k')\}$ type link.

In the binary-dynamics model, each node takes one of two states: active or inactive. An inactive node of which a neighboring node is active changes its status to active stochastically. The dynamics of binary state of nodes can be regarded as a percolation process. The probability that an inactive (i, k) node of which m neighboring nodes are active changes status to active is defined by $RF_i(m, k)$. $RF_i(m, k)$ is called response function. Note that we can change the way of percolation by configuring only $RF_i(m, k)$.

In the binary-dynamics model, when the probability that an (i, k) node is active at time step t

is denoted by $q_k^i(t)$, the probability that a neighbor node of an inactive (i,k) node is active at time step t is given by

$$\bar{q}_k^i(t) = \frac{\sum_{i',k'} P_{k,k'}^{i,i'} q_{k'}^{i'}(t)}{\sum_{i',k'} P_{k,k'}^{i,i'}}. \tag{4.1}$$

Then, $q_k^i(t)$ is given by

$$\begin{aligned} q_k^i(t+1) &= \rho_k^i(0) + (1-\rho_k^i(0)) \sum_{m=0}^{k-1} \binom{k-1}{m} \\ &\times (\bar{q}_k^i(t))^m (1-\bar{q}_k^i(t))^{k-1-m} RF_i(m,k) \\ q_k^i(0) &= \rho_k^i(0), \end{aligned} \tag{4.2}$$

where $\rho_k^i(0)$ is the ratio of the number of active (i,k) nodes to all (i,k) nodes at the initial step. The ratio of the number of active (i,k) nodes to all (i,k) nodes at step $(t+1)$ is then given by

$$\begin{aligned} \rho_k^i(t+1) &= \rho_k^i(0) + (1-\rho_k^i(0)) \sum_{m=0}^{k} \binom{k}{m} \\ &\times (\bar{q}_k^i(t))^m (1-\bar{q}_k^i(t))^{k-m} RF_i(m,k). \end{aligned} \tag{4.3}$$

From the above, the ratio of the number of active nodes to all nodes at step t, denoted as $\rho(t)$, is given by

$$\rho(t) = \sum_i \frac{\sum_{i',k,k'} \frac{P_{k,k'}^{i,i'}}{k}}{\sum_{i,i',k,k'} \frac{P_{k,k'}^{i,i'}}{k}} \rho^i(t) \tag{4.4}$$

$$\text{where } \rho^i(t) = \sum_k \frac{\sum_{i',k'} \frac{P_{k,k'}^{i,i'}}{k}}{\sum_{i',k,k'} \frac{P_{k,k'}^{i,i'}}{k}} \rho_k^i(t). \tag{4.5}$$

In percolation, the ratio of the active nodes to all nodes at which the dynamics (i.e, the iterative calculation of equation (4.1), (4.2) and (4.4)) converge describes the size of the giant component. Thus, $\rho(t)$ for $t \to \infty$ describes the size of the giant component consisting of active nodes when the dynamics converges.

It is true that $P_{k,k'}^{i,i'}$ for the newly created network can be easily calculated given the adjacency matrix of the network after inter-module links addition. However, the $P_{k,k'}^{i,i'}$ derived from an adjacent matrix after inter-module links addition denotes only one example and we only get the robustness of the adjacent matrix. In this strategy, we need to try the whole connection patterns of inter-module links to make topology robust and it is only one result in the situation. This can result in tremendous overhead. In contrast, our method can get the expected robustness of the topology classified according to the strategy of inter-module links addition. Then, our approach narrow the candidates of the link addition strategy to get the robust topology with low overhead when the number of nodes is large. Therefore, our approach can show the policy to make topology robust according to the link addition strategy.

4.3.2 Deriving link probability distribution after addition of inter-module links

To investigate differences in robustness depending on the connection patterns between modules by using the binary-dynamics model, we analyze site percolation when the connection patterns between modules are changed and the probability distribution of links within each module is given. However, we need to consider the change in the probability distribution of links within each module according to addition of inter-module links.

In this paper, we consider that two previously isolated modules are connected by newly created a fixed number of inter-module links, and that these links connect a number of nodes with a fixed degree k in module i to a number of nodes with a fixed degree k' in module i' such that the degrees of nodes that receive the inter-module links is raised from k to d or from k' to d'.

First, we define the probability distribution of links within a module before addition of inter-module links tensor $[Prev_{a,b}^{i,i}]$ in which each element represents the probability that a randomly chosen link is an $\{(i,a),(i,b)\}$ type link. Because we use the probability distribution of links between modules as the target value, we define the probability distribution of links between modules tensor $[Target_{a,b}^{i,i'}]$ in which each element represents the probability that a randomly chosen link is an $\{(i,a),(i',b)\}$ type link. Because tensor $[Prev_{a,b}^{i,i}]$ is for intra-module links and tensor $[Target_{a,b}^{i,i'}]$ is for inter-module links, the conditions of equation (4.6) are satisfied.

$$
\begin{aligned}
Prev_{a,b}^{i,i'} &= 0, \quad \text{for} \quad i \neq i' \\
Target_{a,b}^{i,i'} &= 0, \quad \text{for} \quad i = i'.
\end{aligned} \tag{4.6}
$$

We define Λ_{intra} as the number of links within modules and Λ_{inter} as the number of links between modules. Then, we configure the tensors so as to satisfy the following equations.

$$
\begin{aligned}
&\sum_{i,i',a,b} Prev_{a,b}^{i,i'} + \sum_{i,i',a,b} Target_{a,b}^{i,i'} = 1 \\
&\sum_{i,i',a,b} Prev_{a,b}^{i,i'} : \sum_{i,i',a,b} Target_{a,b}^{i,i'} = \Lambda_{intra} : \Lambda_{inter}.
\end{aligned} \tag{4.7}
$$

We consider the conditions specified by equations (4.6) and (4.7) to calculate the probability distribution of links after addition of inter-module links, denoted by tensor $[Sub_{a,b}^{i,i'}]$, by using $[Prev_{a,b}^{i,i'}]$ and $[Target_{a,b}^{i,i'}]$. Therefore, $Sub_{a,b}^{i,i'}$ is given by:

$$
Sub_{a,b}^{i,i'} = \begin{cases} Prev_{a,b}^{i,i'} + \Delta Prev_{a,b}^{i,i'} & \text{where } i = i' \\ Target_{a,b}^{i,i'} & \text{otherwise,} \end{cases} \tag{4.8}
$$

where $\Delta Prev_{a,b}^{i,i'}$ denotes the amount of change of the probability distribution of links, which is what we need to calculate. Then, we use $Sub_{a,b}^{i,i'}$ instead of $P_{k,k'}^{i,i'}$ in equations (4.1), (4.4) and (4.5) for analysis.

In this chapter we consider the case that the number of modules is two and the number of types of inter-module links is one. When the type of inter-module links is $\{(i,d),(i',d')\}$, it gives the following equation.

$$
Target_{a,b}^{i,i'} = \begin{cases} \frac{\Lambda_{inter}}{\Lambda_{intra}+\Lambda_{inter}} & \text{where } a = d, b = d' \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}
$$

Focusing on module i, if we consider that the degree resulting from the addition of inter-module

links is d, then we add $(d-k)$ links to some (i,k) nodes. Then, $Sub^{i,i}_{k,b}$ is smaller than or equal to $Prev^{i,i}_{k,b}$, and $Sub^{i,i}_{a,k}$ is smaller than or equal to $Prev^{i,i}_{a,k}$. Simultaneously, $Sub^{i,i}_{d,b}$ is larger than or equal to $Prev^{i,i}_{d,b}$ and $Sub^{i,i}_{a,d}$ is larger than or equal to $Prev^{i,i}_{a,d}$. These relations can be hold because we first configure $Prev^{i,i}_{a,b}$ and $Target^{i,i'}_{a,b}$so as to satisfy equations (4.7). This configuration enable us to assume that some amount of the probability of $Prev^{i,i}_{a,k}$ ($Prev^{i,i}_{k,b}$) moves to $Prev^{i,i}_{a,d}$ ($Prev^{i,i}_{d,b}$) when the inter-module links are added in the way mentioned above because some degree-k nodes change to degree-d nodes.

For example, we assume we use the topology shown in Figure 4.1 and add an inter-module link to a degree-one node. In this example, $Prev^{1,1}_{a,b}$, $Target^{1,2}_{a,b}$ and $Sub^{1,1}_{a,b}$ are shown in equations (4.10), (4.11) and (4.12) respectively. Then, we need to derive equation (4.12) from equations (4.10) and (4.11).

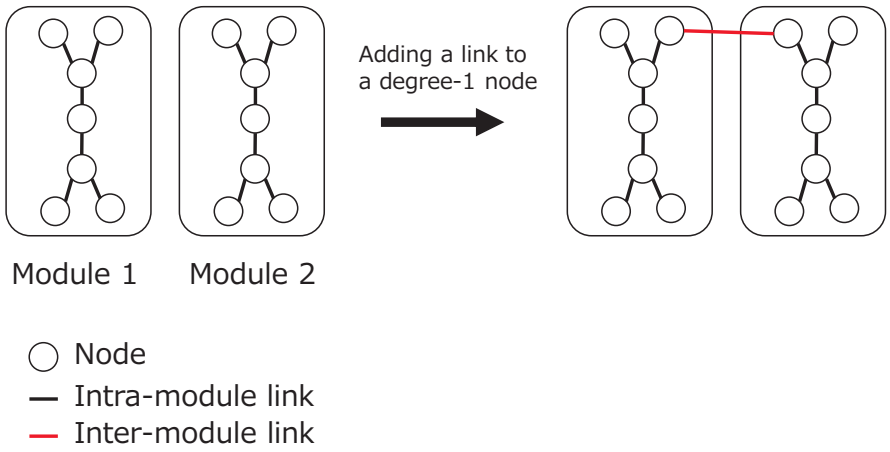


Figure 4.1: Example of inter-module link addition

$$Prev^{1,1}_{a,b} = \frac{1}{26}\begin{pmatrix} 0 & 0 & 4 \\ 0 & 0 & 2 \\ 4 & 2 & 0 \end{pmatrix} \begin{matrix} b=1,2,3, \\ a=1 \\ a=2 \\ a=3 \end{matrix} \tag{4.10}$$

$$Target_{a,b}^{1,2} = Target_{b,a}^{2,1} = \begin{cases} \frac{1}{26} & \text{where } a = 2, b = 2 \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

$$Sub_{a,b}^{1,1} = \frac{1}{26} \begin{array}{c} b = 1, 2, 3, \\ \begin{pmatrix} 0 & 0 & 3 \\ 0 & 0 & 3 \\ 3 & 3 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ a = 1 \\ a = 2 \\ a = 3 \end{array} \tag{4.12}$$

Next, we define the magnitude of influence by link addition. Here, the magnitude of influence means the ratio of the number of intra-module links which change their type because of the change of degree of their endpoint nodes. The magnitude of influence by link addition to a degree-k node is k times as large as that by link addition to a degree-1 node because the number of links changing their type is proportional to the degree of the node that receives the inter-module links. Also, the magnitude of influence by addition of $(d-k)$ links to a degree-k node is $\frac{1}{d-k}$ times as large as that by addition of one link to a degree-k node because the number of links changing their type is proportional to the inverse of the number of links added to one node. Therefore, in the case when the number of created inter-module links is fixed and these inter-module links are connected only to nodes with a fixed degree k such that their degree is raised to d, we can define the magnitude of influence of adding $(d-k)$ links to some degree-k nodes, denoted by $Inf(d,k)$, as follows:

$$Inf(d,k) = H(d-k)\frac{k}{(d-k)} Target_{d,d'}^{i,i'}, \tag{4.13}$$

where H is the Heaviside step function.

Moreover, when we add some links to a degree-k node, the probability that degree of its neighbor is k' is $\frac{Prev_{k,k'}^{i,i}}{\sum_{k'} Prev_{k,k'}^{i,i}}$. This probability is equivalent to the probability that $\{(i,k),(i,k')\}$ link changes its type to $\{(i,d),(i,k')\}$ when we add $(d-k)$ links to a degree-k node. When we consider the above, the amount of change of the probability distribution of links, denoted by $\Delta Prev_{a,b}^{i,i}(d,k)$,

can be calculated by the following equations.

$$\begin{aligned}\Delta Prev^{i,i}_{a,b}(d,k) = &-\delta_{a,k}\frac{Prev^{i,i}_{k,b}}{\sum_{k'} Prev^{i,i}_{k,k'}} \cdot Inf(d,k) \\ &- \delta_{b,k}\frac{Prev^{i,i}_{a,k}}{\sum_{k'} Prev^{i,i}_{k,k'}} \cdot Inf(d,k) \\ &+ \delta_{a,d}\frac{Prev^{i,i}_{k,b}}{\sum_{k'} Prev^{i,i}_{k,k'}} \cdot Inf(d,k) \\ &+ \delta_{b,d}\frac{Prev^{i,i}_{a,k}}{\sum_{k'} Prev^{i,i}_{k,k'}} \cdot Inf(d,k),\end{aligned} \tag{4.14}$$

where δ is the Kronecker delta function.

When we calculate $\Delta Prev^{1,1}_{a,b}(2,1)$ for the example shown in Figure 4.1, $\Delta Prev^{1,1}_{1,3}(2,1) = \Delta Prev^{1,1}_{3,1}(2,1) = -\frac{1}{26}$, $\Delta Prev^{1,1}_{2,3}(2,1) = \Delta Prev^{1,1}_{3,2}(2,1) = \frac{1}{26}$ and the others equal to zero. This is consistent with equation (4.12). In another example shown in Figure 4.2, we can get different probability distribution according to the connected node. We assume an inter-module link is added to a degree-one node. $Prev^{1,1}_{a,b}$ and $Target^{1,2}_{a,b}$ are shown in equations (4.15) and (4.16) respectively. In this case, $\{(1,1),(1,3)\}$ link changes its type to $\{(1,2),(1,3)\}$ and $\{(1,3),(1,1)\}$ link changes its type to $\{(1,3),(1,2)\}$ with probability $\frac{2}{3}$. Also, $\{(1,1),(1,2)\}$ link changes its type to $\{(1,2),(1,2)\}$ and $\{(1,2),(1,1)\}$ link changes its type to $\{(1,2),(1,2)\}$ with probability $\frac{1}{3}$. $Sub^{1,1}_{a,b}$ is shown in equation (4.17) for the former case and in equation (4.18) for the latter case. We average them and can obtain the expected probability distribution shown in equation (4.19). When we calculate $\Delta Prev^{1,1}_{a,b}(2,1)$ for this example, the result is consistent with equation (4.19).

$$\begin{array}{c} b = 1,2,3, \\ Prev^{1,1}_{a,b} = \frac{1}{18}\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \begin{array}{l} a=1 \\ a=2 \\ a=3 \end{array} \end{array} \tag{4.15}$$

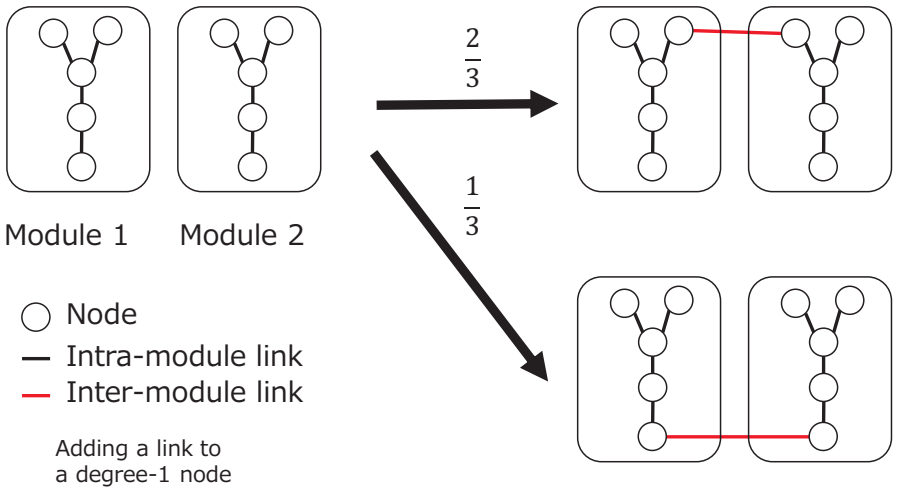


Figure 4.2: Example of link addition in which different probability distribution arises according to the connected node

$$Target_{a,b}^{1,2} = Target_{b,a}^{2,1} = \begin{cases} \frac{1}{18} & \text{where } a = 2, b = 2 \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

$$Sub_{a,b}^{1,1} = \frac{1}{18} \begin{array}{c} b = 1,2,3, \\ \left(\begin{array}{ccc} 0 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 2 & 0 \end{array} \right) \end{array} \begin{array}{l} \\ a = 1 \\ a = 2 \\ a = 3 \end{array} \tag{4.17}$$

$$Sub_{a,b}^{1,1} = \frac{1}{18} \begin{array}{c} b = 1,2,3, \\ \left(\begin{array}{ccc} 0 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 0 \end{array} \right) \end{array} \begin{array}{l} \\ a = 1 \\ a = 2 \\ a = 3 \end{array} \tag{4.18}$$

$$Sub^{1,1}_{a,b} = \frac{1}{54} \begin{pmatrix} 0 & 2 & 4 \\ 2 & 2 & 5 \\ 4 & 5 & 0 \end{pmatrix} \begin{matrix} b = 1,2,3, \\ a = 1 \\ a = 2 \\ a = 3 \end{matrix} \tag{4.19}$$

However, the case of adding links to multiple nodes is not considered in equation (4.14) because the equation cannot describe the change of link type from $\{(i,k),(i,k)\}$ to $\{(i,d),(i,d)\}$ even though it can describe the change from $\{(i,k),(i,b)\}$ to $\{(i,d),(i,b)\}$ and from $\{(i,a),(i,k)\}$ to $\{(i,a),(i,d)\}$. Our method cannot be directly applied to the example shown in Figure 4.3. We assume two inter-module links are added to two degree-2 nodes respectively. In this example, $Prev^{1,1}_{a,b}$ and $Target^{1,2}_{a,b}$ are shown in equation (4.20) and (4.21) respectively. The expected probability distribution after inter-module link addition is shown in equation (4.22). When we calculate $\Delta Prev^{1,1}_{a,b}(3,2)$ for this example, equation (4.23) are derived and $Sub^{1,1}_{3,3}$ remains zero.

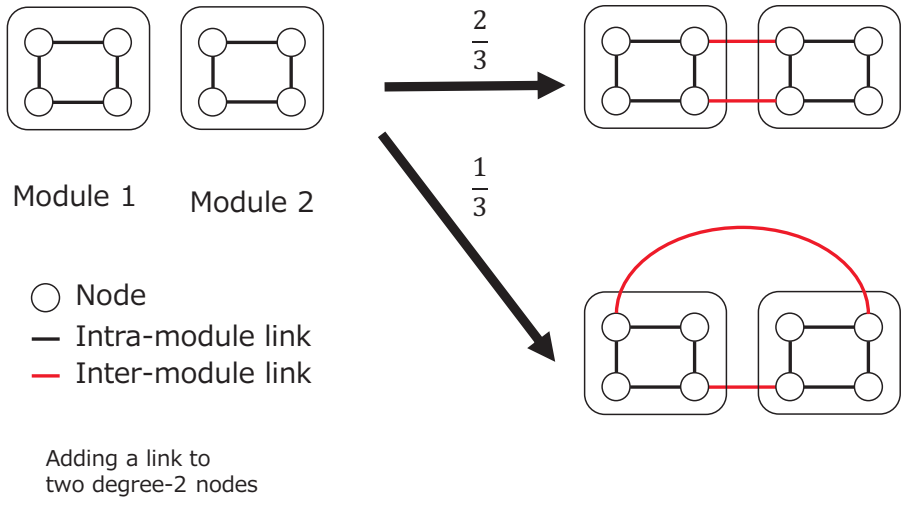


Figure 4.3: Example of multiple links addition in which different probability distribution arises according to the connected node

$$Prev^{1,1}_{a,b} = \frac{1}{20}\begin{pmatrix} 8 & 0 \\ 0 & 0 \end{pmatrix} \begin{matrix} b = 2,3, \\ a = 2 \\ a = 3 \end{matrix} \tag{4.20}$$

$$Target^{1,2}_{a,b} = Target^{2,1}_{b,a} = \begin{cases} \frac{2}{20} & \text{where } a = 3, b = 3 \\ 0 & \text{otherwise.} \end{cases} \tag{4.21}$$

$$Sub^{1,1}_{a,b} = \frac{1}{60}\begin{pmatrix} 4 & 8 \\ 8 & 4 \end{pmatrix} \begin{matrix} b = 2,3, \\ a = 2 \\ a = 3 \end{matrix} \tag{4.22}$$

$$Sub^{1,1}_{a,b} = \frac{1}{20}\begin{pmatrix} 0 & 4 \\ 4 & 0 \end{pmatrix} \begin{matrix} b = 2,3, \\ a = 2 \\ a = 3 \end{matrix} \tag{4.23}$$

This problem can be solved by replacing $Inf(d,k)$ with $\frac{Inf(d,k)}{\Omega^i_k(d)}$ and applying the equation (4.14) $\Omega^i_k(d)$ times, where $\Omega^i_k(d)$ is the number of (i,k) nodes selected as boundary nodes (i.e., endpoint nodes of inter-module links) and connected by $(d-k)$ inter-module links. This configuration enable our method to derive equation (4.22) for the example shown in Figure 4.3.

To calculate $\Omega^i_k(d)$, we calculate the total number of inter-module links between (i,d) nodes and (i',d') nodes after the link addition, denoted by $\Omega^{i,i'}_{d,d'}$. This is given as follows.

$$\begin{aligned} \Omega^{i,i'}_{d,d'} &= \frac{1}{2} N z_{prev} \frac{(Target^{i,i'}_{d,d'} + Target^{i',i}_{d',d})}{\sum_{i,i',a,b} Prev^{i,i'}_{a,b}} \\ &= N z_{prev} \frac{Target^{i,i'}_{d,d'}}{\sum_{i,i',a,b} Prev^{i,i'}_{a,b}}, \end{aligned} \tag{4.24}$$

where N is the total number of nodes and z_{prev} is the average degree of the graph before adding

inter-module links. For this, z_{prev} equals $\sum_{i,k} k\psi_k^i$ where ψ_k^i is the ratio of the number of (i,k) nodes to the total number of nodes and can be calculated as follows:

$$\psi_k^i = \frac{\sum_b \frac{Prev_{k,b}^{i,i}}{k}}{\sum_{i,i',a,b} \frac{Prev_{a,b}^{i,i'}}{a}}. \tag{4.25}$$

Note that $\frac{1}{2}Nz_{prev}$ describes the total number of all intra-module links and $\frac{(Target_{d,d'}^{i,i'} + Target_{d',d}^{i',i})}{\sum_{i,a,b} Prev_{a,b}^{i,i}}$ describes the ratio of the number of inter-module links to intra-module links that connect degree-d nodes from one module to degree-d' nodes of another module to the number of intra-module links. The second equality in equation (4.24) is satisfied because our target is an undirected graph. When we add all inter-module links to some degree-k nodes and add $(d-k)$ inter-module links to each of them, $\Omega_k^i(d)$ can be calculated by the following equation.

$$\Omega_k^i(d) = \frac{\Omega_{d,d'}^{i,i'}}{d-k}. \tag{4.26}$$

Therefore, when the expected total number of nodes is given, $\Omega_k^i(d)$ can be calculated.

4.3.3 Constraints of input variables

In our approach, $[Target_{k,k'}^{i,i'}]$ needs to satisfy some constraints. First, the number of endpoint (i,k) nodes of inter-module links must be less than the number of (i,k) nodes. Second, the number of endpoint (i,k) nodes of inter-module links must be more than $(d'-k')$ and the number of endpoint (i',k') nodes of inter-module links must be more than $(d-k)$ because we assume that multiple links between a pair of nodes are not allowed. Therefore, the constraints can be described as follows:

$$\begin{aligned} (d'-k') \leq \Omega_k^i(d) \leq N\psi_k^i \\ (d-k) \leq \Omega_{k'}^{i'}(d') \leq N\psi_{k'}^{i'}. \end{aligned} \tag{4.27}$$

We can rewrite the above into a constraint on $Target^{i,i'}_{d,d'}$ as follows:

$$\begin{aligned}(d-k)(d'-k')\frac{\sum_{i,a,b} Prev^{i,i}_{a,b}}{N z_{prev}} &\leq Target^{i,i'}_{d,d'} \\ \leq \min\left((d-k)\psi^i_k, (d'-k')\psi^{i'}_{k'}\right)\frac{\sum_{i,a,b} Prev^{i,i}_{a,b}}{z_{prev}}&. \end{aligned} \tag{4.28}$$

Because the lower bound depends on N, there are cases where the desired graph cannot be constructed if N is small.

4.4 Simulation evaluation

We investigate robustness according to the connection patterns of inter-module links by using our proposed method. To evaluate validity of our proposal, we compare the analysis results with the numeric results. In numeric simulations, we evaluate the site percolation process on a graph constructed by a configuration model according to $[Prev^{i,i}_{a,b}]$, $[Target^{i,i'}_{a,b}]$ and N.

The method for constructing a graph based on $[Prev^{i,i}_{a,b}]$, $[Target^{i,i'}_{a,b}]$ and N is listed as follows.

1. Constructing a graph within each module
   (a) Calculating a degree distribution of each module from $[Prev^{i,i}_{a,b}]$ and assigning degree to each node
   (b) Calculating the number of each type of intra-module links from $[Prev^{i,i}_{a,b}]$ and N
   (c) Selecting a pair of endpoint nodes of each intra-module link from the candidates at random and connecting them
2. Adding inter-module links
   (a) Calculating the number of the specified type of inter-module links from the ratio of Λ_{inter} to Λ_{intra}
   (b) Determining a set of the candidates for the boundary nodes at random according to the connection pattern of inter-module links

(c) Selecting a pair of boundary nodes of each inter-module link from the candidates at random and connecting them

In numeric simulations, we assume that inactive nodes are failed nodes.

4.4.1 Percolation on graph ensembles with a given probability distribution

Simulation settings

We assume that the number of modules is two and the probability distributions of intra-module and inter-module links are given by equations (4.29) and (4.30) respectively. Equation (4.29) describes that the degree distribution of each module is uniform. In this case, the ratio of the number of intra-module links to inter-module links is 200 to 3. The expected total number of nodes is 1000 when we analyze the percolation process using our proposed method. We analyze the site percolation process with various combinations of d, d', k and k'.

$$Prev^{i,i}_{a,b} = \frac{1}{812} \begin{array}{c} \begin{array}{ccccc} b=2, & 3, & 4, & 5, & 6 \end{array} \\ \begin{pmatrix} 4 & 6 & 8 & 10 & 12 \\ 6 & 9 & 12 & 15 & 18 \\ 8 & 12 & 16 & 20 & 24 \\ 10 & 15 & 20 & 25 & 30 \\ 12 & 18 & 24 & 30 & 36 \end{pmatrix} \end{array} \begin{array}{l} a=2 \\ a=3 \\ a=4 \\ a=5 \\ a=6 \end{array} \tag{4.29}$$

$$Target^{1,2}_{a,b} = Target^{2,1}_{b,a} = \begin{cases} \frac{6}{812} & \text{where } a = d, b = d' \\ 0 & \text{otherwise.} \end{cases} \tag{4.30}$$

In the percolation process, we use two modes of node removal: random failure and targeted attack. In random failure mode, a removal node is selected uniformly at random, while in targeted attack mode, a removal node is selected in order of decreasing degree. We then define a response function for each node removal mode in order to analyze the site percolation process using our

method. In the binary-dynamics model [49], the response function for the site percolation is defined as follows:

$$RF_i(m,k) = \begin{cases} 0 & \text{where } m = 0 \\ Q_k^i & \text{otherwise,} \end{cases} \tag{4.31}$$

where Q_k^i is the occupation probability of (i,k) nodes. When we assume $(1-\mu)$ of all nodes have failed, then Q_k^i is defined by equation (4.32) for random failure mode [49] and by equation (4.33) for targeted attack mode.

$$Q_k^i = \mu \tag{4.32}$$

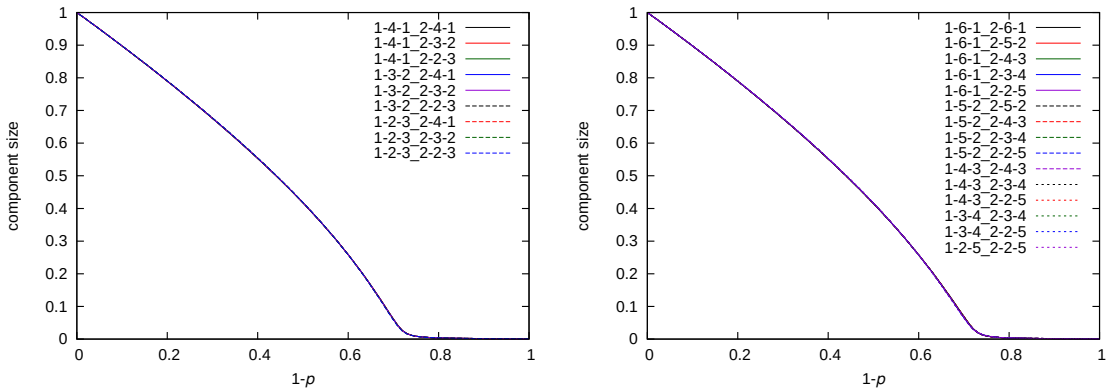
$$Q_k^i = \begin{cases} 1 & \text{where } \sum_{l=1}^{k} \sum_{i} \xi_l^i \leq \mu \\ \dfrac{\mu - \sum_{l=1}^{k-1} \sum_{i} \xi_l^i}{\sum_{i} \xi_k^i} & \text{where } \left(\sum_{l=1}^{k-1} \sum_{i} \xi_l^i < \mu < \sum_{l=1}^{k} \sum_{i} \xi_l^i \right) \\ 0 & \text{otherwise.} \end{cases} \tag{4.33}$$

where ξ_l^i is the ratio of the number of (i,l) nodes to the total number of nodes after the inter-module links addition and can be calculated from $Sub_{a,b}^{i,i'}$. $\sum_{l=1}^{k-1} \sum_{i} \xi_l^i$ describes the ratio of the number of nodes that have lower degree than k to the total number of nodes. Therefore, the first line of equation (4.33) means that all nodes that have lower degree relative to the value of p are occupied. The second line means that degree-k nodes are occupied with the probability which equals to the ratio of active degree-k nodes to all degree-k nodes. The third line means that all nodes that have higher degree relative to the value of p are not occupied.

Here, we consider that analytical results are obtained by iterative calculation of equations (4.1), (4.2) and (4.4).

Analysis of percolation in random failure mode

The results of analysis in random failure mode is shown in Figure 4.1. We evaluate the percolation with various combinations of d, d', k and k'. The results for the case of $d = 5$ and $d' = 5$ are shown in Figure 4.1(a) and the results for the case of $d = 7$ and $d' = 7$ are shown in Figure 4.1(b). Each legend shows the connection pattern of inter-module links and formatted as i-k-$(d-k)$_i'-k'-$(d'-k')$. In each figure, the horizontal axis shows the ratio of the failed nodes, denoted by $(1-\mu)$, and the vertical axis shows the size of the giant component. We assume that the ranking of the robustness of networks is equivalent to the ranking of the size of the giant component at each p value.



(a) $d = 5, d' = 5$ (b) $d = 7, d' = 7$

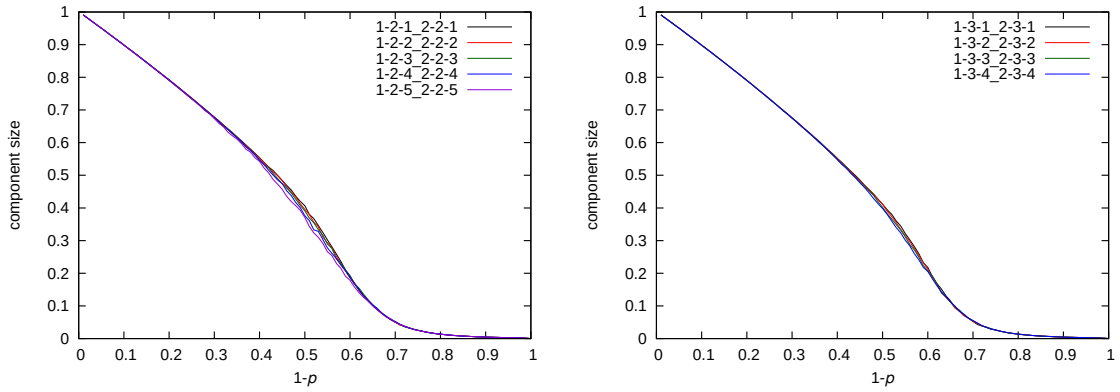
Figure 4.1: Results of percolation analysis in random failure mode

As shown in Figure 4.1, there is little difference depending on the connection pattern of inter-module links in random failure mode.

To evaluate validity of our analytical results showed in Figure 4.1, we construct a graph with the number of nodes set at 1000 and investigate the site percolation process on it in random failure mode. The number of trials is 500. Figure 4.2 shows the results of the site percolation process in random failure mode with k and k' fixed. Each figure shows that there is little difference but the graph in which the number of boundary nodes is small has a slightly more vulnerable structure.

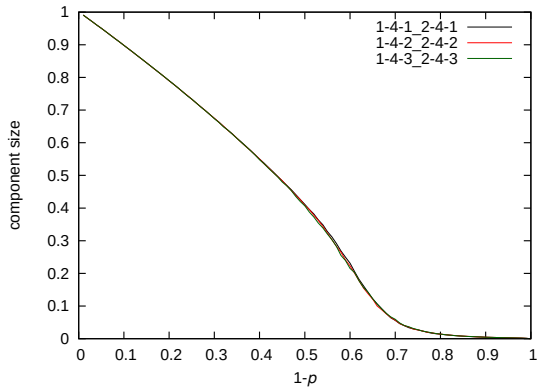
Because the graph becomes divided into modules when the boundary nodes are removed, the graph in which the number of boundary nodes is large has a slightly more robust connectivity.

To compare the graph in which the number of boundary nodes is the same, we show the results of the site percolation process in random failure mode with $(d-k)$ and $(d'-k')$ fixed in Figure 4.3. There are small differences in each of the figures even though the number of boundary nodes is the same. This difference arises from differences in the number of neighbors of boundary nodes. Because the graph is divided into isolated when all neighbors of boundary nodes are removed, the graph in which the average number of intra modular connections of the boundary nodes is large has a slightly more robust connectivity.

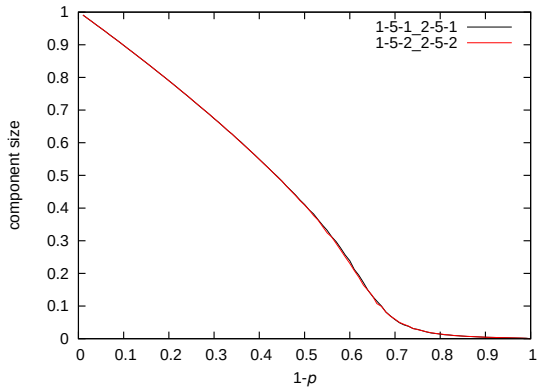


(a) $k=2, k'=2$

(b) $k=3, k'=3$

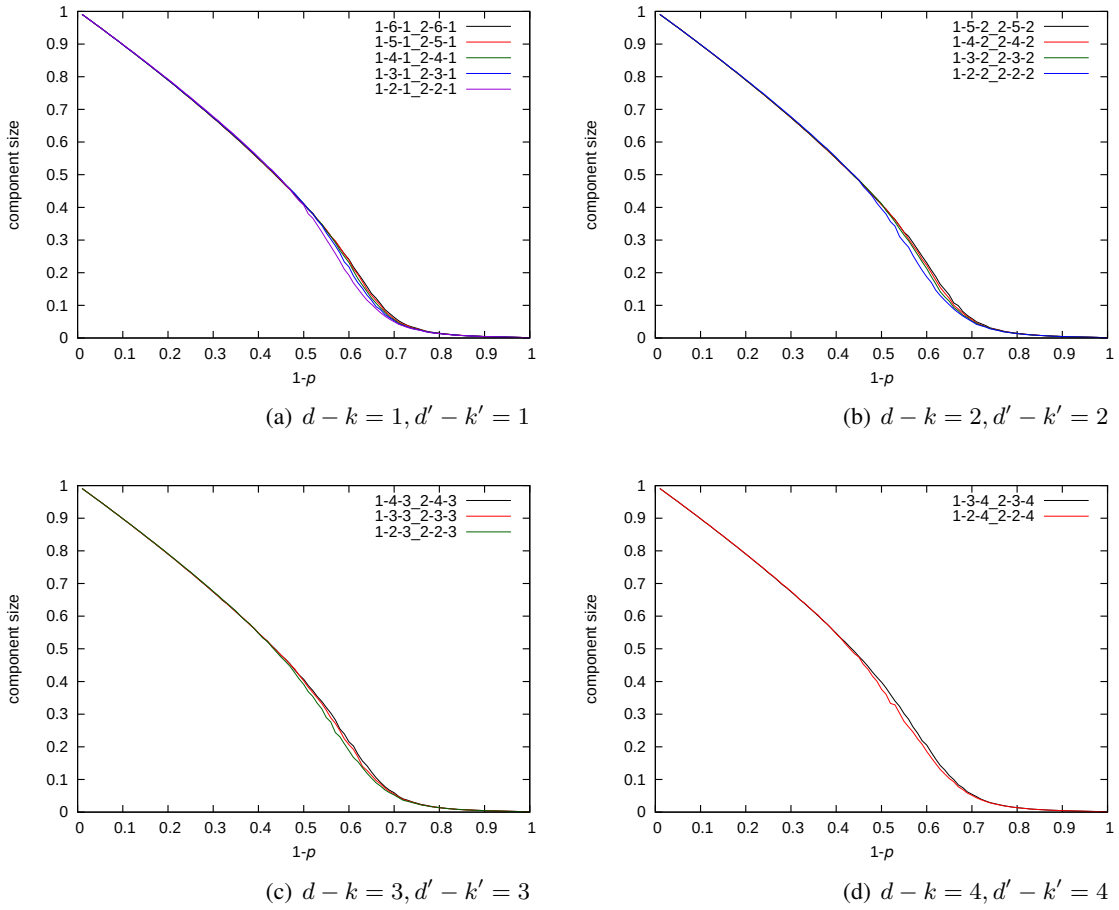


(c) $k=4, k'=4$



(d) $k=5, k'=5$

Figure 4.2: Results of the site percolation process on graphs in random failure mode with k and k' fixed



(a) $d-k=1, d'-k'=1$ (b) $d-k=2, d'-k'=2$

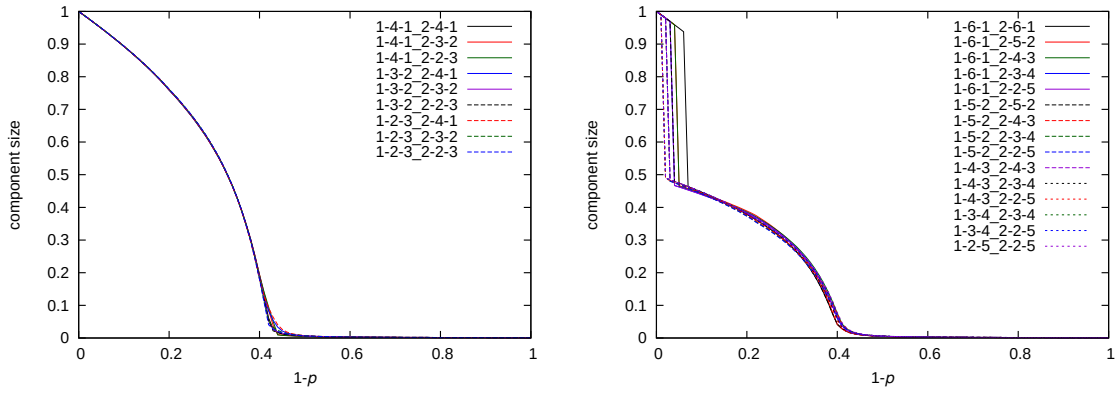
(c) $d-k=3, d'-k'=3$ (d) $d-k=4, d'-k'=4$

Figure 4.3: Results of the site percolation process on graphs in random failure mode with $(d-k)$ and $(d'-k')$ fixed

Analysis of percolation in targeted attack mode

The results of analysis in targeted attack mode are shown in Figure 4.4. The results for the case of $d=5$ and $d'=5$ are shown in Figure 4.4(a) and the results for the case of $d=7$ and $d'=7$ are shown in Figure 4.4(b).

In Figure 4.4(a), all results show that the size of the giant component decreases sharply at $(1-\mu) \simeq 0.4$ and there is little difference between them. Because nodes are removed in order of decreasing degree, all degree-6 and degree-5 nodes are removed when $(1-\mu)$ is larger than 0.4.



(a) $d = 5, d' = 5$ (b) $d = 7, d' = 7$

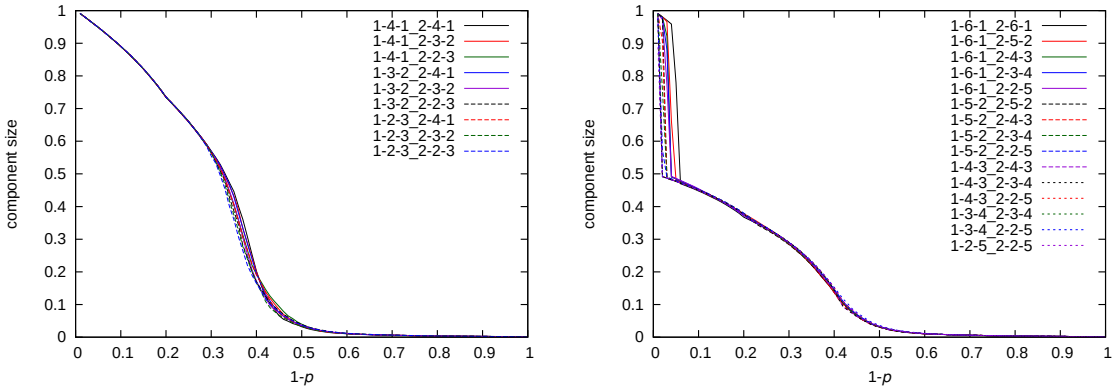
Figure 4.4: Results of percolation analysis in targeted attack mode

In this evaluation, the fragmentation of a graph within a module occurs before the removal of all inter-module links because d and d' are set to 5.

In Figure 4.4(b), every result shows a phase transition at small $(1 - \mu)$, and the size of the giant component decreases gradually after that. In this evaluation, a phase transition indicates the division of the graph into modules by removal of all inter-module links. Because the maximum degree is 7 when d and d' equal 7, connectivity of the graph makes it vulnerable to targeted attack. In addition, robustness of connectivity is different according to k and k'. The larger $\Omega_k^i(d)$ means the graph has more robust connectivity because modules are connected until removal of all (i, d) or (i', d') nodes. In this evaluation, because all $(1, 7)$ nodes and all $(2, 7)$ nodes are the boundary nodes and a removal node is selected uniformly and randomly from degree-7 nodes at small $(1 - \mu)$, the larger $(\Omega_k^1(7) + \Omega_{k'}^2(7))$ means the graph has more robust connectivity. Therefore, the graph with larger $(\frac{1}{d-k} + \frac{1}{d-k'})$ has more robust connectivity. Note that the targeted attack we use in this evaluation is based on the nodes' degrees. This means these results can be different according to the definition of Q_k^i.

To evaluate validity of our analytical results showed in Figure 4.4, we construct a graph with the number of nodes set at 1000 and investigate the site percolation process on it in targeted attack mode. The number of trials is 500. Figure 4.5 shows the results of the site percolation process in targeted attack mode. By comparison with Figure 4.4, although the giant component size in each

method is slightly different, the order of robustness of each connection pattern is the same. From these results, analytical results are in good agreement with numerical results and the graph in which the number of boundary nodes is large has more robust connectivity when we use the targeted attack mode in which a removal node is selected in order of decreasing degree.



(a) $d = 5, d' = 5$ (b) $d = 7, d' = 7$

Figure 4.5: The results of the site percolation process on graph in targeted attack mode

Applicable range of our proposed method

When we consider the IoT environment, there may be cases where the number of nodes belonging to each module is large. Given this context, we need to investigate the applicable range of our proposed method for making policies according to the results of analysis.

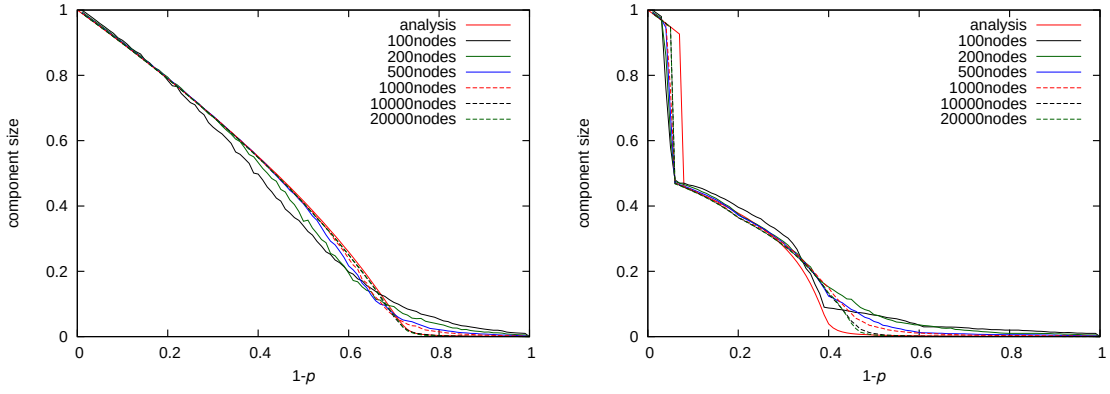
To show the applicable range of our proposed method, we compare the results of analysis to the results of the percolation process on the graph when the number of nodes is one of 100, 200, 500, 1000, 10000 and 20000 respectively. We use the method of 1-6-1_2-6-1 as an example, the number of trials is 100 and the expected number of nodes for analysis is 1000.

Figure 4.6(a) shows the results in random failure mode and Figure 4.6(b) shows the results in targeted attack mode. Both show that difference between analysis and numeric becomes larger as the number of nodes becomes smaller. This means that it is hard to apply our proposal to a graph in which the number of nodes is small because our approach derives average properties of random graph ensembles. However, such a small graph can be analyzed by numeric simulations. Therefore,

this is not a problem for our proposal whose target is the graphs in which the number of nodes is large.

In Figure 4.6(b), however, the results of analysis cannot completely capture any of the numeric results. In this evaluation, degree-7 nodes are 6% of the total nodes and half of them belong to module 1 and the rest belong to module 2. In the analysis method, the selection of a removal node is regarded as completely uniform. This means that the graph is not divided into modules until all degree-7 nodes are removed. In the numeric simulations, however, the graph can be divided into modules when half of degree-7 nodes are removed because removal of all degree-7 nodes belonging to module 1 or 2 results in fragmentation of the graph. Therefore, the numeric results show a phase transition at an earlier stage than the analysis. In addition, Figure 4.6(b) shows that a graph with a small number of nodes becomes vulnerable because the number of degree-7 nodes is small and the graph fragments easily.

The analytic results cannot completely capture any of the numeric results in targeted attack mode because of the reasons described above. The result for 10000 nodes is almost the same as for 20000 nodes, which means that the difference between numeric and analysis cannot get any closer. Because this difference comes from the difference of an order of node that fails in the numeric simulation, the way that nodes fail is also an important factor which determines whether analytic results completely capture numeric results or not.



(a) random failure mode (b) targeted attack mode

Figure 4.6: Comparison of the result of analysis and numeric

4.4.2 Percolation on graph ensembles with a probability distribution derived from a given intra-module topology

The results shown above is obtained when we first give a probability distribution and investigate the site percolation process on a graph constructed by the configuration model to evaluate validity of our proposed method. When we consider an actual situation, however, we need to show a policy to make a graph robust by connecting multiple existing networks.

Simulation settings

In this part, we investigate robustness of a graph in which modules are constructed by Erdős-Rényi (ER) model, Barabási-Albert (BA) model and Watts-Strogatz (WS) model [75] respectively. We use the parameters shown in Table 4.1 for constructing a module. By using these parameters, the expectation of the average degree in a module is six when the number of nodes in a module is 500. The number of modules is two, the number of nodes in a module is 500 and the number of links added between modules is 1% of the total number of links within modules. We compare the numerical results for the site percolation process on a graph constructed by modules and inter-module links, with the analytical results using the probability distribution obtained from existing modules.

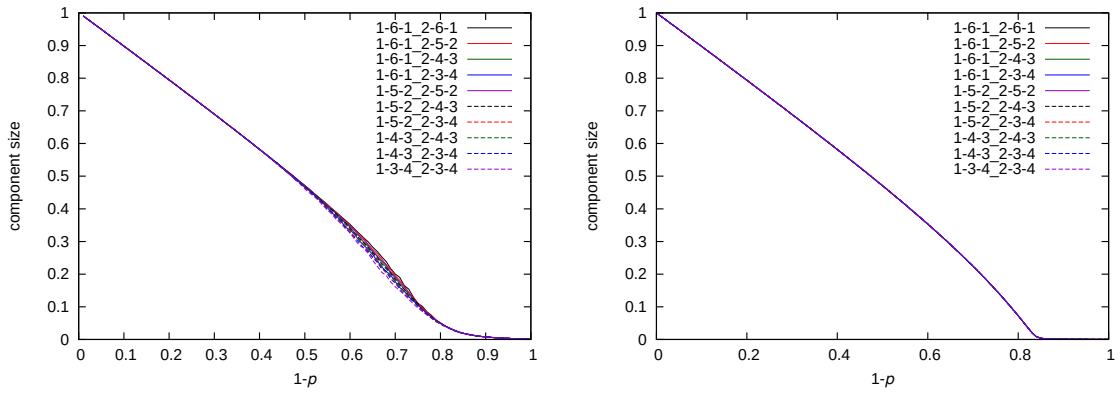
Table 4.1: Parameters for constructing a topology within a module

model	parameter	value
ER	$p_{(ER)}$	0.012
BA	$m_{0(BA)}$	7
	$m_{(BA)}$	6
WS	$m_{(WS)}$	6
	$\beta_{(WS)}$	0.01

Results by using ER model

The numerical and analytical results for the random failure mode in a network composed of ER modules are shown in Figure 4.7. The number of trials for numeric in each topology is 100. The minimum degree within a module is one and the maximum degree within a module is 15.

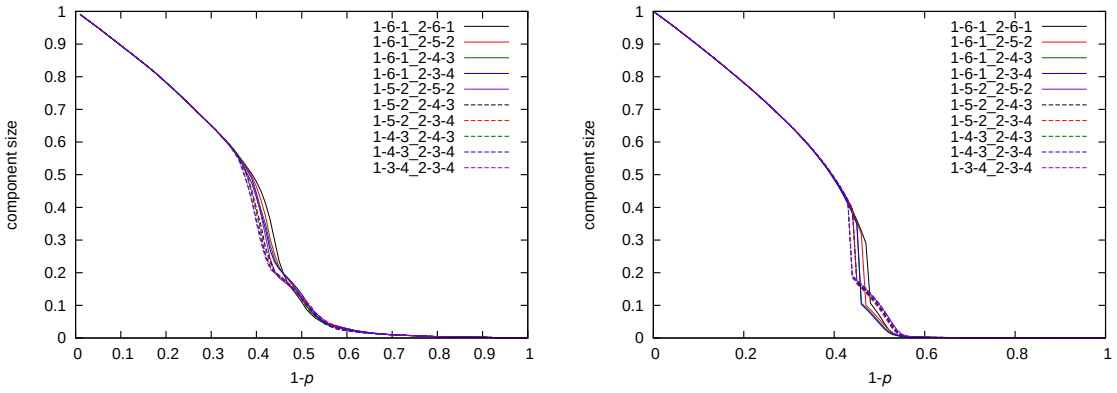
The results of numeric and analysis for the case of $d = 7$ and $d' = 7$ are shown in Figure 4.7(a) and Figure 4.7(b) respectively. All of them are consistent with the results shown above. For the random failure mode, the analytical results depend only slightly on the connection pattern on inter-module links. In numeric, the graph in which the number of boundary nodes is small has a slightly vulnerable structure.



(a) numeric (b) analysis

Figure 4.7: The results in random failure mode (ER model)

When we use the ER model for constructing a topology within a module, the results of numeric and analysis in targeted attack mode are shown in Figure 4.8. The results in Figure 4.8(b) are consistent with the results shown above in terms of the first sudden decay of the size of the giant component. After the fragmentation of the network, the ranking of the size of the giant component changes. This is because the connectivity of the intra module network remains high after the fragmentation when k (or k') is set to small value. When we set k and d to 6 and 7 respectively, little number of degree-6 nodes remain after all degree-7 nodes are removed and degree-6 nodes are crucial for the connectivity of the intra module because of their high degrees. This leads to the vulnerable connectivity of intra module network after the fragmentation. In these results, the connectivity of the module 1 is larger than that of module 2. Therefore, k is more dominant than k' after the fragmentation. Figure 4.8(a) shows that the order of robustness of each connection pattern is the same as the results of analysis although the giant component sizes are not completely the same.



(a) numeric (b) analysis

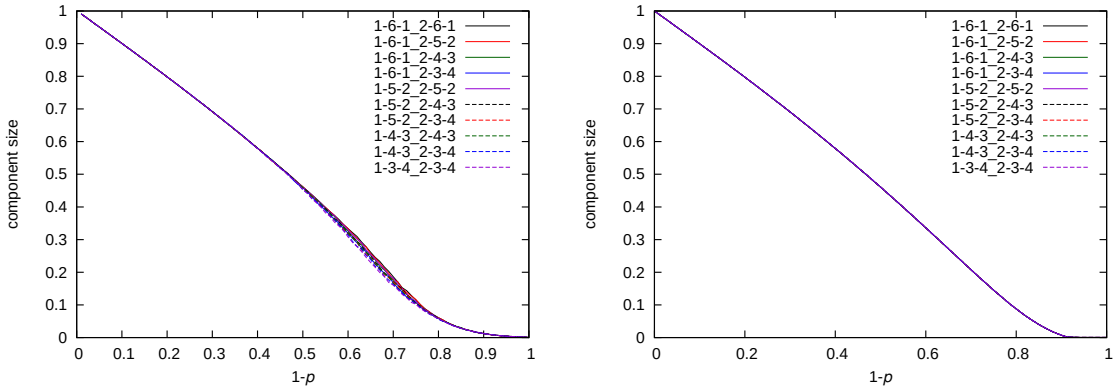
Figure 4.8: The results in targeted attack mode (ER model)

Results by using BA model

The numerical and analytical results for the random failure mode in a network composed of BA modules are shown in Figure 4.9. The number of trials for numeric in each topology is 100. The minimum degree within a module is three and the maximum degree within a module is 76.

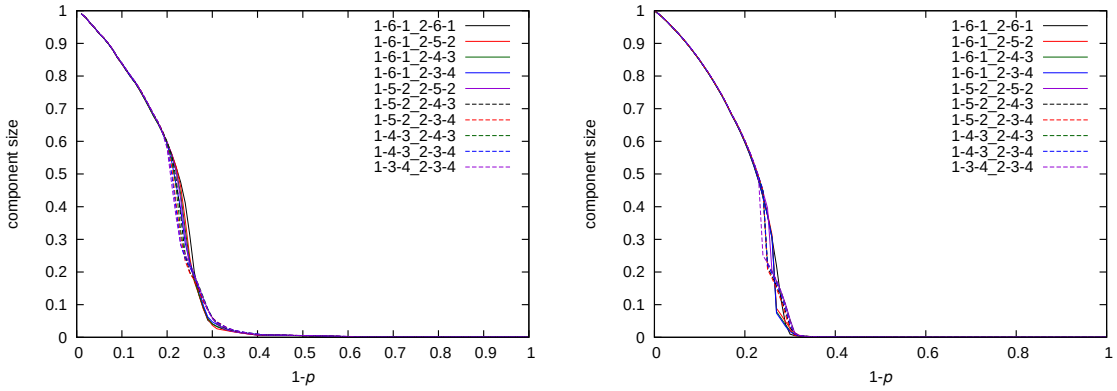
The results of numeric and analysis for the case of $d = 7$ and $d' = 7$ are shown in Figure 4.9(a) and Figure 4.9(b) respectively. These results show that there is little difference depending on the connection pattern of inter-module links in random failure mode. Because we use BA model for constructing an intra-module graph, it has a power-law degree distribution. Therefore, difference of robustness will not occur unless we set d and d' to greatly high value.

When we use the BA model for constructing a graph within a module, the results of numeric and analysis in targeted attack mode are shown in Figure 4.10. The results of numeric and analysis for the case of $d = 7$ and $d' = 7$ are shown in Figure 4.10(a) and Figure 4.10(b) respectively. Although the sizes of the giant components are almost same, the graph in which the number of boundary nodes is large has a slightly higher robust connectivity before the fragmentation of the network in both analysis and numeric. After the fragmentation of the network, the ranking of the sizes of the giant components changes. The reason for this is the same as that mentioned in Section 4.4.2.



(a) numeric (b) analysis

Figure 4.9: The results in random failure mode (BA model)



(a) numeric (b) analysis

Figure 4.10: The results in targeted attack mode (BA model)

Results by using WS model

The numerical and analytical results for the random failure mode in a network composed of WS modules are shown in Figure 4.11. The number of trials for numeric in each topology is 100. The minimum degree within a module is four and the maximum degree within a module is eight.

The results of numeric and analysis for the case of $d = 7$ and $d' = 7$ are shown in Figure 4.11(a) and Figure 4.11(b) respectively. Each figure shows that the giant component size in each method is almost same. However, the giant component size in numeric decreases at an earlier step compared with the results of analysis. This is because the graph constructed by the WS model has a ring-shaped structure and can fragment easily in numeric while the theory assumes random intra-module connections.

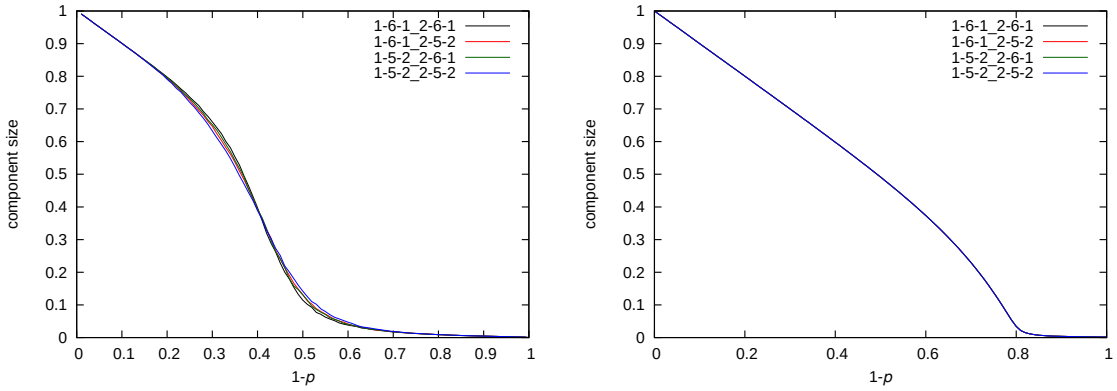
When we use the WS model for constructing a graph within a module, the results of numeric and analysis in targeted attack mode are shown in Figure 4.12. From the results shown in Figure 4.12(a) and Figure 4.12(b), the results of analysis are consistent with the results shown above. A graph constructed by method in which $(\frac{1}{d-k} + \frac{1}{d-k'})$ is large has high robustness. After a graph fragments into modules, however, the giant component size in numeric decreases at an earlier step compared with the results of analysis because of the same reason discussed in random failure mode.

From these results, our proposal can capture the order of robustness according to the time step of fragmentation of a graph. However, because we do not consider the structural properties of a graph within a module, a difference of analysis and numeric occurs when a graph within a module has a special structure. It should be future work how to resolve this problem.

4.5 Summary

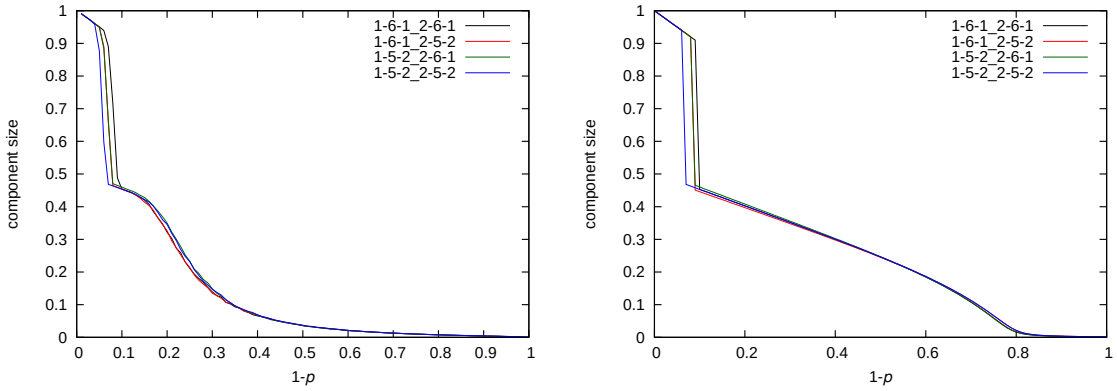
In this chapter, we propose a method for estimating robustness of graph ensembles after addition of inter-module links when the probability distribution of links within a module is given. Our proposal is for a binary-dynamics model [49] and add a new tool for the model. We investigate robustness according to the connection patterns of inter-module links.

In our simulation evaluation, we compare the robustness of the networks in various connection patterns of inter-module links when we fixed the values d and d' that describe the degrees of the



(a) numeric (b) analysis

Figure 4.11: The results in random failure mode (WS model)



(a) numeric (b) analysis

Figure 4.12: The results in targeted attack mode (WS model)

boundary nodes after the link addition. Simulation experiments showed that graphs have robust connectivity when the number of nodes selected as boundary nodes and the degrees of the boundary nodes before the link addition are large. Moreover, a constructed topology when we set d to small value has large robust connectivity. These results are consistent with the results shown in Chapter 3. In Chapter 3, we evaluate the robustness and adaptivity of a VWSN topology when we use a module whose size is small. In this chapter, we show that the same results can be gained in analysis using various module sizes. This means that we can verify that our proposal in Chapter 3 is an effective way of constructing inter-module links for making a topology robust. To evaluate validity of the analysis results, we evaluate the percolation process on a graph constructed by a configuration model and find that the analysis results are in agreement with the numeric results. Moreover, we investigate the applicable range of our proposed method and showed that the difference between the analysis and numeric results increases as the number of nodes decreases.

To show a policy to make a graph robust by connecting multiple existing networks, we investigate robustness of a graph composed of modules with a given internal structure. The results show that our proposal can capture the ranking of the robustness according to the positions of percolation transition (sudden decay of the size of the giant component). However, because we do not consider the structural properties of a graph within a module, a difference of analysis and numeric occurs when a graph within a module has a special structure. It should be future work how to resolve this problem.

For further investigation, we want to consider addition of multiple types of inter-module links. When multiple types of inter-module links are added, we cannot ignore the order of addition of the inter-module links because the conditional probability of the probability distribution of links changes after each addition of an inter-module link. This analysis will be realized when we use equation (4.14) multiple times according to the sequence of the type of new added link. Therefore, our method can also be applied to the graph in which multiple module exist.

Chapter 5

Conclusion

In the near future, billions of devices are expected to be connected to the Internet and to communicate with one another for various applications, thereby creating an Internet of Things (IoT) environment. To achieve the IoT environment, WSNs will be a critical technology for allowing collecting and acting on environmental information. In WSNs, the main problems concern operation in an energy-scarce, harsh wireless environment and managing a large number of devices. In this vein, energy efficiency, reliability, and scalability have been addressed for decades, and methods limited to specific situations have been proposed. Within the IoT environment, changes to traffic patterns, variation in traffic demand, and addition or removal of sensor nodes can all occur. As a result, providing stable applications remains difficult, and most existing research has not considered IoT-specific situations. Reliability, even in the face of such environmental change, is important. Therefore, in this thesis, we provide an architecture for reliable IoT applications.

In Chapter 2, we first propose an any-to-any routing protocol possessing scalability and adaptivity against environmental change. To this end, we take potential-based routing that offers scalability, adaptivity, and energy efficiency and extend it to any-to-any communication. We achieve potential-based any-to-any routing (PBAR) by merging potential-based upstream and downstream routing. In PBAR, multiple sink nodes construct independent potential fields, and all nodes have a set of potentials that is used as a virtual coordinate. We defined virtual distance by using these virtual coordinates and use the result as a routing metric. Through OMNeT++ simulation, we evaluated the

data delivery ratio and path stretch for various node densities, as well as robustness against failure of multiple sensor nodes or a sink node. PBAR achieves a data delivery ratio greater than 99.7% when the network has a suitable node density. Even if multiple sensor nodes fail or a sink node fails, the data delivery ratio recovers immediately after sensor node failure or sink node failure.

In Chapter 3, we next consider how to integrate multiple heterogeneous WSNs and how reliable sensor resources could be leased to application developers. For future IoT environments, all devices will be connected to the Internet and will communicate with one another to meet the demands of various applications. For WSNs to serve as communication infrastructure of the future IoT, integrating deployed sensor resources will be a critical consideration. In this study, we show an overall architecture that is suitable for constructing and running virtual wireless sensor network (VWSN) services within a VWSN topology as one strategy for the integration of WSNs. Our approach to constructing the VWSN topology was inspired by brain networks, which possess a reliable structure. We conduct a simulation of the practical situation to evaluate the reliability of our proposed VWSN topology in an actual environment. The results of simulation experiments showed that reliability of the VWSN topology constructed by BICM (ll,LL) was highest against a targeted attack. However, the structure of the physical topology under the VWSN topology had a critical impact on reliability in terms of connectivity.

From the studies mentioned above, we found that a connection pattern between modules is critical for robust connectivity and stability of services. Therefore, to improve reliability of the VWSN topology mentioned above, we use an analytical approach and propose a method for estimating the robustness of graph ensembles after addition of inter-module links when the probability distribution of links within a module is as given in Chapter 4. Our proposal is based on a binary-dynamics model [49], and is an extension of the model. We investigated robustness according to the connection patterns of inter-module links. Simulation experiments showed that graphs have robust connectivity when inter-module links connect lower-degree nodes and the number of endpoint nodes of inter-module links is high. This finding is consistent with the above study. In addition, to evaluate validity of the analysis results, we evaluated the percolation process on a graph constructed by a configuration model and found that the analysis results agreed with the numeric results. Moreover,

we investigated the applicable range of our proposed method, and showed that the difference between the analysis and numeric results widened as the number of nodes decreased. To demonstrate a policy to increase robustness of a graph by connecting multiple existing networks, we investigated robustness of a graph when we initially provided the graph within a module. The results showed that our proposal can characterize the degree of robustness according to the time step of fragmentation of a graph.

Future architectures for the IoT must have high tolerance for general purposes so that application developers can deploy various applications without tight coupling to substrates. Therefore, virtualization of physical networks and sensor substrates is of increasing significance, and it is becoming increasingly important to perform substantially more investigation of how to integrate and manage heterogeneous networks for stability of services.

Future work entails further research in exploring architecture that is evolvable in response to environmental changes. An evolvable architecture should not only configure behavior adaptively to environmental changes but also restructure its framework for each application. Analyzing patterns and histories of the environmental changes should enable us to recalculate and reassign each application resource so as to construct more reliable virtual networks according to the situation.

Bibliography

[1] C. Sarkar, A. U. N. SN, R. V. Prasad, A. Rahim, R. Neisse, and G. Baldini, "DIAT: A scalable distributed architecture for IoT," *IEEE Internet of Things Journal*, vol. 2, no. 3, pp. 230–239, Jun. 2015.

[2] J. A. Stankovic, "Research directions for cyber physical systems in wireless and mobile healthcare," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 1, p. 1, Nov. 2016.

[3] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, "Internet-of-Things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 10–16, Oct. 2016.

[4] L. Hou, S. Zhao, X. Xiong, K. Zheng, P. Chatzimisios, M. S. Hossain, and W. Xiang, "Internet of Things Cloud: Architecture and implementation," *CoRR*, vol. abs/1609.07712, 2016. [Online]. Available: http://arxiv.org/abs/1609.07712

[5] H. Wang, S. Chen, H. Xu, M. Ai, and Y. Shi, "SoftNet: A software defined decentralized mobile network architecture toward 5G," *IEEE Network*, vol. 29, no. 2, pp. 16–22, March 2015.

[6] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments," *CoRR*, vol. abs/1606.02007, 2016. [Online]. Available: http://arxiv.org/abs/1606.02007

[7] E. Natalizio and V. Loscri, "Controlled mobility in mobile sensor networks: advantages, issues and challenges," *Telecommunication Systems*, vol. 52, no. 4, pp. 2411–2418, Apr. 2013.

[8] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[9] Q. Liang, D. Yao, S. Deng, S. Gong, and J. Zhu, "Potential field based routing to support QoS in WSN," *Journal of Computer Information Systems*, vol. 8, no. 6, pp. 2375–2385, 2012.

[10] P. T. A. Quang and D.-S. Kim, "Enhancing real-time delivery of gradient routing for industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 61–68, Feb. 2012.

[11] H. Bandara, A. P. Jayasumana, and T. H. Illangasekare, "A top-down clustering and cluster-tree-based routing scheme for wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2011, no. 2011, pp. 1–17, Mar. 2011.

[12] K. Xu, Y. Qu, and K. Yang, "A tutorial on the Internet of Things: from a heterogeneous network integration perspective," *IEEE Network*, vol. 30, no. 2, pp. 102–108, Mar. 2016.

[13] I. Khan, R. Jafrin, F. Z. Errounda, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "A data annotation architecture for semantic applications in virtualized wireless sensor networks," in *Proceedings of 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, May 2015, pp. 27–35.

[14] S. Quadri and O. Sidek, "Software maintenance of deployed wireless sensor nodes for structural health monitoring systems," *International Journal of Computer Engineering Science*, vol. 3, no. 2, p. 1, Feb. 2013.

[15] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz, "GARUDA: Achieving effective reliability for downstream communication in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 2, pp. 214–230, Feb. 2008.

[16] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica, "Beacon vector routing: Scalable point-to-point routing in wireless sensornets," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, May 2005, pp. 329–342.

[17] S. Duquennoy, O. Landsiedel, and T. Voigt, "Let the tree bloom: scalable opportunistic routing with ORPL," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, Nov. 2013, p. 2.

[18] S. Guo, L. He, Y. Gu, B. Jiang, and T. He, "Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2787–2802, Nov. 2014.

[19] S. Fauji and K. Kalpakis, "A gossip-based energy efficient protocol for robust in-network aggregation in wireless sensor networks," in *Proceedings of 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, Mar. 2011, pp. 166–171.

[20] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Tech. Rep., Oct. 2003.

[21] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," Tech. Rep., Jul. 2003.

[22] L. Shu, Y. Zhang, L. T. Yang, Y. Wang, M. Hauswirth, and N. Xiong, "TPGF: geographic routing in wireless multimedia sensor networks," *Telecommunication Systems*, vol. 44, no. 1-2, pp. 79–95, Jun. 2010.

[23] A. Caruso, S. Chessa, S. De, and A. Urpi, "GPS free coordinate assignment and routing in wireless sensor networks," in *Proceedings of the 24th IEEE Annual Joint Conference on Computer and Communications Societies.*, vol. 1. IEEE, Mar. 2005, pp. 150–160.

[24] M.-J. Tsai, H.-Y. Yang, B.-H. Liu, and W.-Q. Huang, "Virtual-coordinate-based delivery-guaranteed routing protocol in wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 4, pp. 1228–1241, Aug. 2009.

[25] J. Zhou, Y. Chen, B. Leong, and B. Feng, "Practical virtual coordinates for large wireless sensor networks," in *Proceedings of the 18th IEEE International Conference on Network Protocols (ICNP)*. IEEE, Oct. 2010, pp. 41–51.

[26] S. Xia, X. Yin, H. Wu, M. Jin, and X. D. Gu, "Deterministic greedy routing with guaranteed delivery in 3D wireless sensor networks," *Axioms*, vol. 3, no. 2, pp. 177–201, May 2014.

[27] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by DHTs," in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, Oct. 2006, pp. 351–362.

[28] A. Awad, R. German, and F. Dressler, "Exploiting virtual coordinates for improved routing performance in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1214–1226, Sep. 2011.

[29] C. Wu, R. Yuan, and H. Zhou, "A novel load balanced and lifetime maximization routing protocol in wireless sensor networks," in *Proceedings of 2008 IEEE Vehicular Technology Conference (VTC Spring 2008)*. IEEE, May 2008, pp. 113–117.

[30] H. Yoo, M. Shim, and D. Kim, "A scalable multi-sink gradient-based routing protocol for traffic load balancing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–16, Sep. 2011.

[31] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, "Controlled potential-based routing for large-scale wireless sensor networks," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*. ACM, Nov. 2011, pp. 187–196.

[32] M. M. Islam, M. M. Hassan, G.-W. Lee, and E.-N. Huh, "A survey on virtualization of wireless sensor networks," *Sensors*, vol. 12, no. 2, pp. 2175–2207, Feb. 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/2/2175

[33] I. Ishaq, J. Hoebeke, I. Moerman, and P. Demeester, "Internet of Things virtual networks: Bringing network virtualization to resource-constrained devices," in *Proceedings of IEEE International Conference on Green Computing and Communications (GreenCom 2012)*, Nov. 2012, pp. 293–300.

[34] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: early architecture and research perspectives," *Network, IEEE*, vol. 29, no. 3, pp. 104–112, May 2015.

[35] W. Wang, S. De, G. Cassar, and K. Moessner, "An experimental study on geospatial indexing for sensor service discovery," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3528 – 3538, May 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095741741400757X

[36] P. Levis and D. Culler, "MatÉ: A tiny virtual machine for sensor networks," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*. ACM, Dec. 2002, pp. 85–95.

[37] R. Chu, L. Gu, Y. Liu, M. Li, and X. Lu, "SenSmart: Adaptive stack management for multi-tasking sensor networks," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 137–150, Jan. 2013.

[38] C.-L. Fok, G.-C. Roman, and C. Lu, "Agilla: A mobile agent middleware for self-adaptive wireless sensor networks," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 3, pp. 16:1–16:26, Jul. 2009.

[39] S. Toyonaga, D. Kominami, M. Sugano, and M. Murata, "Potential-based downstream routing for wireless sensor networks," in *Proceedings of International Conference on Systems and Networks Communications (ICSNC 2012)*. IARIA, Nov. 2012, pp. 59–64.

[40] ——, "Potential-based routing for supporting robust any-to-any communication in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–13, Dec. 2013.

[41] S. Toyonaga, D. Kominami, and M. Murata, "Brain-inspired method for constructing a robust virtual wireless sensor network," in *Proceedings of the 10th International Conference on Computing and Network Communications (CoCoNet 2015)*, Dec. 2015, pp. 69–75.

[42] ——, "Virtual wireless sensor networks: Adaptive brain-inspired configuration for Internet of Things," *Sensors*, vol. 16, no. 8, p. 1323, Aug. 2016.

[43] D. Papo, J. M. Buldú, S. Boccaletti, and E. T. Bullmore, "Complex network theory and the brain," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 369, no. 1653, p. 20130520, Dec. 2014.

[44] D. Meunier, R. Lambiotte, and E. T. Bullmore, "Modular and hierarchically modular organization of brain networks," *Frontiers in Neuroscience*, vol. 4, no. 200, pp. 1–11, Dec. 2010.

[45] M. E. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, no. 2 Pt 2, p. 026118, Jul. 2001.

[46] E. a. Leicht and R. M. D'Souza, "Percolation on interacting networks," *ArXiv e-prints*, vol. 2, p. 5, Jul. 2009. [Online]. Available: http://arxiv.org/abs/0907.0894

[47] B. Min, S. D. Yi, K. M. Lee, and K. I. Goh, "Network robustness of multiplex networks with interlayer degree correlations," *Physical Review E*, vol. 89, no. 4, pp. 1–9, Apr. 2014.

[48] G. Dong, J. Gao, R. Du, L. Tian, H. E. Stanley, and S. Havlin, "Robustness of network of networks under targeted attack," *Physical Review E*, vol. 87, no. 5, pp. 1–11, May 2013.

[49] S. Melnik, M. A. Porter, P. J. Mucha, and J. P. Gleeson, "Dynamics on modular networks with heterogeneous correlations," *Chaos (Woodbury, N.Y.)*, vol. 24, no. 2, p. 023106, Apr. 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24985420

[50] Y. Noh, U. Lee, P. Wang, B. S. C. Choi, and M. Gerla, "VAPR: void-aware pressure routing for underwater sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 5, pp. 895–908, May 2013.

[51] C. Damdinsuren, D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, "Lifetime extension based on residual energy for receiver-driven multi-hop wireless network," *Cluster Computing*, vol. 16, no. 3, pp. 469–480, Sep. 2013.

[52] A. Varga, "Omnet++," in *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg, 2010, pp. 35–59. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12331-3_3

[53] R. Nagel and S. Eichler, "Efficient and realistic mobility and channel modeling for vanet scenarios using omnet++ and inet-framework," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Mar. 2008, p. 89.

[54] I. Leontiadis, C. Efstratiou, C. Mascolo, and J. Crowcroft, "SenShare: Transforming sensor networks into multi-application sensing infrastructures," in *Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN 2012)*. Springer Berlin Heidelberg, Feb. 2012, pp. 65–81. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28169-3_5

[55] P. Trakadas, H. Leligou, T. Zahariadis, P. Karkazis, and L. Sarakis, *The Future Internet: Future Internet Assembly 2013: Validated Results and New Horizons*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Managing QoS for Future Internet Applications over Virtual Sensor Networks, pp. 52–63. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38082-2_5

[56] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 553–576, Mar. 2015.

[57] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, Nov. 2013.

[58] P. Rajeshwari, B. Shanthini, and M. Prince, "Hierarchical energy efficient clustering algorithm for WSN," *Middle-East Journal of Scientific Research*, vol. 23, no. sensing, signal processing and security, pp. 108–117, 2015.

[59] M. Demirbas, A. Arora, V. Mittal, and V. Kulathumani, "Design and analysis of a fast local clustering service for wireless sensor networks," in *Proceedings of the 1st International Conference on Broadband Networks (BroadNets 2004)*. IEEE, Oct. 2004, pp. 700–709.

[60] H. Zhang and A. Arora, "GS 3: scalable self-configuration and self-healing in wireless sensor networks," *Computer Networks*, vol. 43, no. 4, pp. 459–480, Nov. 2003.

[61] K. F. Seyed Mahdi Jameii and M. Dehghan, "AMOF: adaptive multi-objective optimization framework for coverage and topology control in heterogeneous wireless sensor networks," *Telecommunication Systems*, vol. 61, no. 3, pp. 515 – 530, Mar. 2016.

[62] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet-of-Things," in *Proceedings of 2014 IEEE network operations and management symposium (NOMS)*. IEEE, May 2014, pp. 1–9.

[63] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDIoT: a software defined based Internet of Things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 4, pp. 453 – 461, Aug. 2015.

[64] E. Bullmore and O. Sporns, "The economy of brain network organization," *Nature Reviews Neuroscience*, vol. 13, no. 5, pp. 336–349, May 2012.

[65] C.-h. Park, S. Y. Kim, Y.-H. Kim, and K. Kim, "Comparison of the small-world topology between anatomical and functional connectivity in the human brain," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 23, pp. 5958–5962, Oct. 2008.

[66] L. K. Gallos, H. A. Makse, and M. Sigman, "A small world of weak ties provides optimal global integration of self-similar modules in functional brain networks," *Proceedings of the National Academy of Sciences*, vol. 109, no. 8, pp. 2825–2830, Feb. 2012.

[67] S. Achard, R. Salvador, B. Whitcher, J. Suckling, and E. Bullmore, "A resilient, low-frequency, small-world human brain functional network with highly connected association cortical hubs," *The Journal of Neuroscience*, vol. 26, no. 1, pp. 63–72, Jan. 2006.

[68] P. E. Vertes, A. F. Alexander-Bloch, N. Gogtay, J. N. Giedd, J. L. Rapoport, and E. T. Bullmore, "Simple models of human brain functional networks," *Proceedings of the National Academy of Sciences*, vol. 109, no. 15, pp. 5868–5873, Mar. 2012. [Online]. Available: http://www.pnas.org/content/109/15/5868.abstract

[69] D. Samu, A. K. Seth, and T. Nowotny, "Influence of wiring cost on the large-scale architecture of human cortical connectivity," *PLoS Computational Biology*, vol. 10, no. 4, p. e1003557, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1371\%2Fjournal.pcbi.1003557

[70] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, Apr. 2006.

[71] R. Agarwal, A. Banerjee, V. Gauthier, M. Becker, C. K. Yeo, and B. S. Lee, "Achieving small-world properties using bio-inspired techniques in wireless networks," *The Computer Journal*, vol. 55, no. 8, pp. 909–931, Mar. 2012.

[72] Q. K. Telesford, K. E. Joyce, S. Hayasaka, J. H. Burdette, and P. J. Laurienti, "The ubiquity of small-world networks," *Brain Connectivity*, vol. 1, no. 5, pp. 367–375, Nov. 2011.

[73] S. Toyonaga, D. Kominami, M. Sugano, and M. Murata, "Potential-based routing for supporting robust any-to-any communication in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–13, Dec. 2013.

[74] S. Hong, C. Lv, T. Zhao, B. Wang, J. Wang, and J. Zhu, "Cascading failure analysis and restoration strategy in an interdependent network," *Journal of Physics A:*

Mathematical and Theoretical, vol. 49, no. 19, p. 195101, Apr. 2016. [Online]. Available: http://stacks.iop.org/1751-8121/49/i=19/a=195101

[75] M. Chopra and M. Madan, "Network analysis by using various models of the online social media networks," *International Journal of Advanced Research in Computer Science*, vol. 6, no. 1, pp. 111–116, Jan. 2015.