Thesis for the Degree of Doctor of Philosophy

Logical Clustering Using Bio-inspired Synchronicity for Wireless Sensor Networks

Yu Niu

Department of Computer Engineering

Graduate School

Kyung Hee University

Seoul, Korea

February, 2012

Logical Clustering Using Bio-inspired Synchronicity for Wireless Sensor Networks

Yu Niu

Department of Computer Engineering

Graduate School

Kyung Hee University

Seoul, Korea

February, 2012

Logical Clustering Using Bio-inspired Synchronicity for Wireless Sensor Networks

by

Yu Niu

Advised by

Professor Sungyoung Lee

Submitted to the Department of Computer Engineering
and the Faculty of the Graduate School of
Kyung Hee University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Dissertation Committee:

Professor Ok-Sam Chae, Ph.D. . . . . .

Professor Young-Koo Lee, Ph.D. . . . . .

Professor Dong Han Kim, Ph.D. . . . . .

Professor Brian J.d'Auriol, Ph.D. . . . . .

Professor Sungyoung Lee, Ph.D. . . . . .

Logical Clustering Using Bio-inspired Synchronicity for Wireless Sensor Networks

by

Yu Niu

Submitted to the Department of Computer Engineering
on January 9th, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Synchronization/desynchronization and clustering are three common and useful primitives for sensor network topology control, information collection, and energy efficiency. However, most of the existing algorithms' clustering criteria are limited to node location information and ignore the other nature and characteristics of the nodes, as well as requirements of applications. Moreover, to achieve efficient communication and data procession in clustered networks, additional techniques such as time synchronization/desynchronization are demanded. Currently, all the node clustering or synchronization/desynchronization algorithms are single functional; none of them can accomplished the three functions at one time.

To address the narrowness on clustering criterion and single-function limitation in existing clustering techniques, this thesis provides an autonomic concurrent General Criteria-based Clustering (GCC) method and the associated data communication framework for wireless sensor networks. The GCC method is based on the Delayed Inhibitory Pulse-Coupling Oscillator (DIPCO) model, which is inspired by the clustering phenomena of the biological neuron dynamic system. By regrading sensor nodes as homogeneous oscillators with different initial phase values, GCC engineers the mathematical model into wireless sensor network to achieve the node clustering goal. In the model, the oscillators' initial phase distribution works as the clustering criterion. Due to the abstractness of phase, any node-related data or properties can be mapped as oscillators' initial phase values so work as the node clustering criteria. This characteristic greatly extends the choice and flexibility of node clustering criteria, thus realizing the generality. With the general clustering criteria, GCC can capture both physical and logical correlations among sensor nodes and represent them as Logical Clusters (LCs). On the other hand, the logical clustering procedure is also the phase synchronizing/desynchronizing procedure. After clustering, with simple transformation, the same cluster members can automatically get time synchronized and different clusters get desynchronized.

With inherent synchronization and desynchronization, a data communication framework is established on the logical clustered network. Among LCs, a quasi-TDMA schedule, supported by desynchronization, helps to distinguish the different cluster members, eliminate the communication interference, and save energy. Inside each LC, a novel Backbone tree structure for data routing is built to accommodate the diverse cluster spatial distribution. The simulation shows that

the GCC method can generate diverse logical clusters and synchronization/desynchronization co-existence with acceptable time and energy usage. Furthermore, using the framework, inter/intra logical cluster data communication can be achieved with a fast response speed, long lifetime, and low interference.

Thesis Supervisor: Sungyoung Lee
Title: Professor

Acknowledgments

Many different people provided help, support, and input that brought this thesis to fruition. Firstly, I would like to express my gratitude to my supervisor, Prof. Sungyoung Lee, who gave me the opportunity to do this Ph.D research and the freedom to try out new ideas. Even during the most difficult periods of my research, his ceaseless understanding and encouragement provided tremendous support.

I would like to express my special thanks to my co-advisor Prof. Brian J.d'Auriol, who continually enlightens me with the proper method of transforming preliminary ideas into effective solutions through independent thinking. In every step of my research, he was always there with invaluable advice and stimulating suggestions. He taught me how to face a multitude problems in my research and paper writing.

I would also like to thank Prof. Jinsung Cho, who advised me in the first two years of my study, and Prof. Young-Koo Lee, whose hardworking spirit and serious research attitude inspired me throughout my period of study.

I am also indebted to my thesis committee whose comments helped me markedly improve the presentation of this thesis. Thanks especially to Prof. Ok-Sam Chae's original interpretation from the perspective of optimization, which provided truly novel insights from an advanced viewpoint.

Most importantly, I want to express my deepest gratitude and appreciation to my family: my parents, whose endless love is my primary motivation to accomplish this work; my elder brother, Li Niu, who guided me into the palace of knowledge since I was a child; and my beloved fiance, Chong Tang, who gives me love, care, and support at all times. This thesis would never have been completed without their constant support and encouragement.

Many thanks to my friends and colleges at the Ubiquitous Computing Lab, the previous Ubiquitous Sensor Network Team, and current Cyber Physical System Team for their collaboration and encouragement over the last several years.

Yu Niu

February 2012

Seoul, Korea

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Wireless sensor network

Due to advances in wireless communications and MEMS (Micro Electro-Mechanical Systems), the development of networks of low-cost, low-power, multi-functional sensors has increasingly generated attention. A sensor node integrates hardware and software for sensing, data processing, and communication. A wireless sensor network (WSN) consists of multiple spatially distributed autonomous sensor nodes often with a limited processing unit where all nodes interact locally and some nodes interact with gateways-termed Base Stations (BS), the latter often having more computational energy and communication resources.

WSNs help to develop a number of applications such as coordinated target detection and localization, surveillance, ubiquitous health care and environmental monitoring. However, no matter the specific purposes to deploy the sensor nodes, the primary job of the networks is to collect information: nodes sense their environment information; after simple local processing, the data packets are sent to the sink node or base station for further processing. To fulfill this duty, the random deployed nodes need to be self-organized into some topology (tree, grid, random flat, cluster, hierarchical structure, etc.), so that so that every node knows its data report routing path. The Media Access Control (MAC) scheme is also needed to schedule the data transmission and

avoid collisions in wireless channels. Furthermore, to get precise and concise data information, the coordinated network time baseline is vital, which guarantees the accuracy of sensed and aggregated data helping to schedule the transmissions. Other than the above mentioned topology control, packet routing, transmission schedule and time synchronization, security and response speed issues of network are also important. All of these concerns are under the special constraints of WSNs, such as limited energy supply, resource limited CPU, memory, bandwidth, randomly deployment, etc.

In this thesis, our research subjects involve the topology control-specifically node clustering-, time synchronization, and data gathering in wireless sensor networks.

1.1.2 Node clustering

In order to send information from a large number of sensor nodes to the base station, it is necessary and economical to group sensors into clusters. Clustering in wireless sensor network is often studied because it is useful for network scaling, routing, and beneficial for energy efficiency. Scaling involves grouping large numbers of nodes into a smaller number of clusters. The head node in each cluster serves as the manager and spokesman, which makes the management of a large scale sensor network much easier. Routing involves member nodes sending the data to their head nodes and the data packets are transmitted only by head nodes to the base station. It avoids the high energy cost on each individual node and the serious interference caused by directly transmitting. Aside from local transmission, energy efficiency also benefits from the clusters' alternation sleeping mechanism when there are no transmission needs.

Most of the existing clustering algorithms in sensor networks group the nodes simply based on their physical distances to the designated cluster head (CH). The only criteria of dividing cluster member nodes are their geographic locations or communication costs, which neglects the nature and characteristics of sensor nodes, as well as requirements of the application level. Moreover, many existing algorithms require additional assistant techniques for communication in clustered

networks. For example, some need a central coordinator or special coding methods (Direct-Sequence Spread Spectrum (DSSS) in LEACH [31]) to avoid collision among nodes and clusters. And some others demand additional time synchronization mechanisms for data aggregation or other recording functions.

1.1.3 Synchronicity/Desynchronicity

Time synchronization is a critical piece of infrastructure in distributed systems and wireless sensor networks that make particularly extensive use of synchronized time. Applications such as target tracking, MAC layer access schedule, time varying data collection, and secure cryptographic schemes need either local or global synchronized time scale as a precondition. When considering a clustering scheme, synchronization and associated transmission scheduling will have a considerable effect on network lifetime and the overall network performance [38].

According to strictness, network synchronization can be categorized into three levels: time synchronization, synchronicity, and desynchronicity. **Time synchronization**, which is implied in most synchronizing situations, means that all local node clocks agree on a global time scale with zero offset and the same clock drift rate. If needed, this network-wide global time can be transferred to the real-world natural clock time. This type of synchronization is usually required by applications that exchange data or commands between sensor networks and external environments, such as target tracking, intrusion detection, and real-time surveillance. **Synchronicity** means that different nodes keep the same activity frequency and instant phase state, but they have no notion of time scale. Given the common coordinate zero point, synchronicity can be upgraded to strict time synchronization through some simple computation. Synchronicity itself, as the network coordination primitive, is also extremely useful for some applications, such as scheduling node sleep/wake up duty cycles, coordinating nodes sampling across multiple nodes, etc. **Desynchronicity** means that although nodes have different activity phase and no notion of time scale, the phase differences (or offsets) between any pair of nodes are non-zero and fixed. Different

from the chaotic asynchronism in which the activities of nodes have no rhythm at all, the regular desynchronized nodes still keep the same acting frequency. The desynchronicity can be regarded as a kind of generalized synchronization. And if the offset values are known, desynchronicity also can be shifted to synchronicity and further to a synchronized time scale. Desynchronization makes nodes active in different time intervals to avoid transmission collision, and balance the working load amongst them. In desynchronized networks, nodes usually interleave periodic events to occur in a round-robin schedule.

Synchronization/desynchronization and clustering are the three most common and useful primitive functions for sensor network application. Each of these techniques has been studied individually, and substantive algorithms or protocols for various goals have been proposed. However, none of them has accomplished the three functions in one algorithm.

1.2 Problem Statement

In respect to the narrowness of clustering criterion and single function limitation in existing clustering techniques, there are three requirements for the new clustering method. First, the node partition criteria should have more choice, more flexibility, and generality. Second, during the network clustering procedure, the application requirement should also be considered. Third, if possible, the multi-functional clustering method could increase working efficiency. This research considers providing an alternative node clustering method that is general-criteria based and multi-functional for sensor networks.

According to the definition of clustering: “Partition a set into subsets (clusters), so that the members in each subset share some similarity, where similarity is often defined as proximity according to some defined distance measure, also called the criterion measure.” [26]. The distance measure in normal network node clustering is the physical one between nodes and their head nodes. While in this thesis, the nodes’ synchronicity and desynchronicity characteristics are used as the ‘distance measure’ to define a new type of cluster: Logical Cluster (LC) (Figure 1.1). In the

Figure 1.1: The relationship between logical cluster, synchronicity/desynchronicity and time synchronization.

same LC, nodes are phase synchronized; and among different LCs, they are phase desynchronized. The LC's construction is realized by the General Criteria-based Clustering (GCC) method. The cluster identification and communication management are supported by the data communication framework.

In GCC, sensor nodes are regarded as homogeneous oscillators and imitate the interactive coupling activities in the neuron dynamic system. By emitting and receiving the pulse-like signals, oscillators adjust their phase values according to different coupling schemes. The clustering procedure is also the phase synchronizing and desynchronizing procedure. In the model, oscillators' initial phase distribution acts as the clustering criterion. Because of the mathematical characteristics of phase, its initial value practically can be mapped from any node-related data or properties, which means that any node-related characteristic data can work as the clustering criteria. This greatly extends the flexibility and possible choice of node clustering criteria. Moreover, with the synchronization and desynchronization property on the logical clustered network, the inter/intra data communication framework provides an energy-efficient, collision-free, scalable infrastructure for the duty of data collection in sensor networks.

The research methodology of this thesis is from theory (Pulse-Coupled Oscillator model) to engineering (General Criteria Clustering method and Data communication framework).

1.3 Contributions

The contributions of this thesis are as follows:

Extend the node clustering criteria choice by proposing a general criteria based clustering

method (GCC), in which any node-related property including application requirement data can work as the clustering criteria.

Provide a multi-functional compound primitive package for sensor networks in which GCC can accomplish the node clustering, synchronizing and desynchronizing simutaneously.

Defines a novel Logical Cluster concept that captures and represents the logical correlation among sensor nodes.

Provide a systematic survey of the sync/desync property of pulse-coupled oscillator networks and their application in the field of computer science.

For the first time, analyze and introduce the dynamics of Delayed Inhibitory Pulse-Coupled Oscillators (DIPCO) in wireless sensor networks.

Propose a quick response, load balanced Backbone tree topology for data routing in arbitrary shaped network field.

1.4 Thesis Organization

The thesis is organized as follows: Chapter 2 provides the general background for node clustering and bio-inspired synchronicity. The Delayed Inhibitory Pulse-Coupled Oscillator (DIPCO) model, as the core theoretical basis of this thesis, is introduced in Section 3.2 of Chapter 3. The converging mechanism of two nodes DIPCO model is attached in Appendix A. Chapter 4 engineers the mathematic model into wireless sensor networks through the method of General-Criteria based Clustering (GCC) to achieve the logical node clustering (LC) results. Taking into account the special characteristics of the logical clustered network, Chapter 5 presents a data communication framework on it. The numerical test, parameter analysis on GCC and the simulation test on the communication framework are given at the ends of both chapters respectively. The summary of the thesis and direction of future research are discussed in Chapter 6. For more details of the thesis organization one section level, please refer to Section 3.1 and Figure 3.1.

Chapter 2

Related work

2.1 Node Clustering

Node clustering organizes the large amount of nodes into a smaller number of groups so as to increase scalability and reduce the complexity of network management. In existing clustering algorithms, given the parameters for variation of energy consumption in the nodes, several questions are considered: How many clusters are needed, how many sensors should be connected to each cluster head (CH), how to select CH, and where each CH should be positioned. The cluster divisions and formations are led by the cluster head. Nodes that have closer distance to the designated head node are grouped together. The only criterion for deciding which cluster a node belongs to is its distance to the head node or the communication cost. The main work in traditional clustering algorithms is choosing proper cluster heads. Some algorithms choose cluster heads based on the residual energy [87], some are based on node connection degree [6, 15, 37, 45] and some are based on the combine weights of several critical factors (energy, mobility, transmission cost, connecting degree, etc.) [16]. All of these algorithms partition the network area geometrically and the correlation between clusters formations and the sensing environment or sensing data are not considered.

2.2 Time Synchronization

For the time synchronization in distributed system, different network configurations have different methods. For wired networks that can connect to the internet, nodes regularly receive or actively require the Coordinated Universal Time (UTC) for synchronization. For wireless networks, if nodes are equipped with Global Positioning System (GPS) receivers, networks can receive standard time information via satellite. Due to environmental and cost issues, it is expensive for wireless sensor networks to get synchronized time from outside. However, for most of applications that do not require communication with the natural world in real-time, the natural clock synchronized network time scale is unnecessary. Therefore, the time synchronization task in a wireless sensor network is mainly to build a common time scale for network nodes' sharing.

Most existing time synchronization methods are 'sender-to-receiver' style - the receiver passively adjusts its local clock to agree with the sender's time scale through the time message. In this mode, two factors will influence the precision of synchronization: time delay during the message transmission procedure and clock drifting. Current protocols take several measures against the precision decrease caused by delay, such as making more than one low layer timestamp at every transmission/receiving step to measure the exact delay value (FTSP [39], DMTS [58]), or using pair-wised messages to calculate the time difference and delay at the same time (TPSN [25], TINY-SYNC and MINI-SYNC [67], etc.). For clock drifting, based on the temporal linear stable characteristics of the clock drift rate, RBS [21] and FTSP use the previous time dataset to predict the future drift rate in a certain period. On the other hand, some protocols are intended not to construct the long lasting global synchronized time scale but only the localized one. They are usually event driven or on-demanded that cause nodes to exchange time information only between the event involved nodes. RITS [64] does the synchronizing along with the data routing procedure simultaneously. Algorithm in [61] converts the time scale node by node. Except for the static network situation, these protocols are also suitable for the dynamic ad hoc networks.

All of the above protocols are devoted to construct a uniformed time scale whether in a local

or global network. As discussed in the opening of this thesis, they are performing time synchronization.

2.3 Bio-synchronicity

In natural systems, a lot of self-organized synchronicity or synchrony phenomena have been observed in which a population of units act collectively but without a notion of time. Examples include crickets chirping, fireflies flashing in unison [13], pacemaking of heart cells [57], circadian pacemaking in networks of neurons [84–86], rhythm of earthquakes [33], and even synchronous clapping in concert audience. Abstracting from different biophysical phenomena and details, these units and other biological neurons can be modeled as phase oscillators with different phase coupling schemes.

2.3.1 Pulse-Coupled Oscillator system

The dynamics of oscillators are typically described in phase space. Through short pulse-like signals (spiking pulse between the neuron oscillators), oscillators interact with each other to adjust their phase values. In detail, the interaction can be described as: in a network, each oscillator has a phase value that increases with time. When the phase reaches a particular threshold, the oscillator fires, and the phase is reset to a lower value. As a consequence of firing, a pulse is sent to several other oscillators. The pulse relays a positive or negative adjustment to the oscillators and this transfer of phase jump defines the coupling in the system. This type of interaction, in which the oscillator communicates by firing sudden pulses, is called pulse-coupled.

Different than the ‘sender-to-receiver’ manner, the oscillators in system are au pair through a specific coupling scheme, the phases of oscillators can agree on a new shared common value - getting synchronized. Since the procedure is performed in a ‘peer-to-peer’ manner, this kind of synchronizing is also called mutual synchronization. The synchronicity of oscillators is the base of synchronized time scale in networks. In reality, the standard International Atomic Time and Uni-

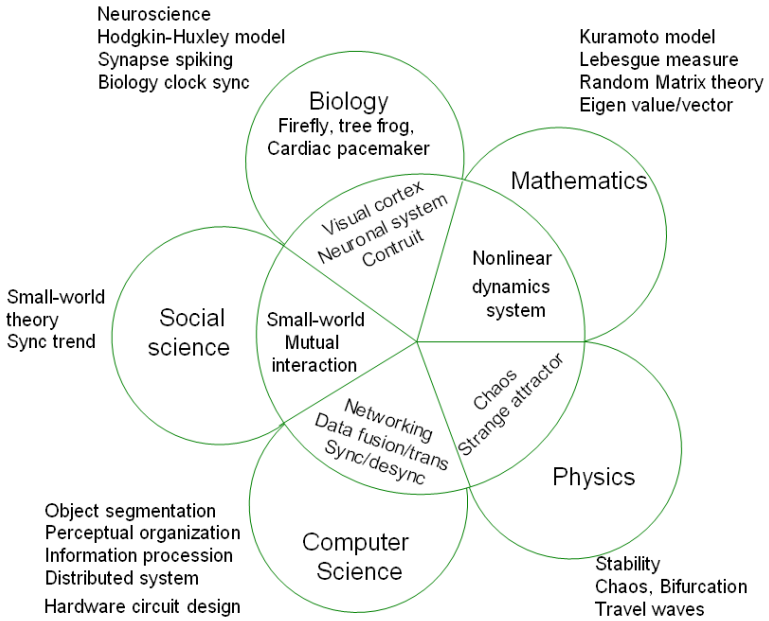


Figure 2.1: Extensive background and application of pulse-coupled Oscillator systems.

versal Coordinated Time (UTC) is built on the oscillating period of the cesium atom (International System of Units (SI) in 1967 defined the second as the duration of 9192631770 cycles of radiation corresponding to the transition between two energy levels of the caesium-133 atom [2]). Also, well-coordinated synchrony within and between neuronal populations appears to be an important mechanisms for neuronal signaling and information processing [47].

Based on different self-oscillating functions and coupling schemes, the pulse-coupled oscillator system may produce various results, and the above mentioned global synchrony is the most noticeable and intriguing one. The research on pulse-coupled oscillator systems involves extensive scientific areas including physics, chemistry, biology, engineering, medicine, and others. Along with the long research history (since 1673 Huygens's observation on two anti-phase synchronized pendulums), biological and mathematical models using mathematics, physics tools, and even rising social science theories have been developed (Figure 2.1).

The following section will introduce some representative, related works on pulse-coupled oscillator systems along the thread of synchronicity and desynchronicity. It will be divided into two parts: mathematic developments and applications in the field of computer science.

2.3.2 Mathematic development of PCO

In a seminal paper, Mirrollo and Strogatz [50] generalize the various oscillator and coupling forms in previous works and focus on the monotonicity and concavity of the oscillating function's curve. They proved that, when the oscillating function is monotonically increase and concave down ($f' > 0$ and $f'' < 0$), the excitatory coupling form will make the phases of all oscillators converge to a global synchronicity, regardless of the number of nodes and their initial states (M-S mode). Although this conclusion is very useful, its solidity is based on three ideal assumptions: 1) all-to-all coupling forms, 2) instantaneous firing and coupling, and 3)homogeneous oscillators. After Mirrollo and Strogatz, a lot of work tries to observe if the conclusion is still tenable when any of these three assumption are relaxed; or what kind of new results will emerge, if some factors of the oscillating and coupling functions are changed.

Follow the method of Mirollo and strogatz, Chen [17] partially removes the temporal all-to-all coupling assumption by taking into account a refractory period (threshold) after the neuron firing, in which no coupling could happen. He proves that for almost all initial conditions, the oscillation system approaches synchrony when the threshold (length of the refractory period) condition is below a critical value. Additionally, Lucarelli and Wang [46] lift the spatial all-to-all communication requirements indicated in [50] and prove that the convergence can be achieved by local coupling. When continuous firing is allowed and no refractory period exists, Hong and Scaglione [36] point out that if the coupling strength and node density are big enough, the 'avalanche effects' will happen and the whole network will lock to synchronicity immediately. The similar 'avalanche' state is discussed in Carl and L. F. Abbott's self-sustained firing pattern work [76], in which the neurons have no intrinsic ability to fire. To keep the system running, at each timestep, an amount of firing

must take place that will excite other neurons to increase their level of activation and ultimately fire themselves, which then makes the system self-sustain the entire firing, activation motion: the result is that the system stays in this 'avalanche' state indefinitely. In the model, because the system is discrete time event driven, the oscillators will not lock together, and instead induce two different converging results: many groups fire sequentially or the whole population breaks into two groups firing alternately. Alexandre and Rodolphe [49] explore the situation of concave-up time-evolution ($f' > 0$ and $f'' > 0$) with excitatory pulse coupling and give a proof of global convergence to a phase-locked clustered behavior. They also observe that as the proportion of non-identical oscillators increase, the oscillators converge from total to a robust intermediate situation of partial clustering, characterized by few oscillators traveling among nearly phase-locked clusters.

The research on pulse-coupled oscillators' non-instantaneous coupling is especially prolific and deep. Whether in the biological nervous system or computer networks, various transmission and reaction delays are inevitable. When adding delay into the pulse-coupled oscillator synchronizing model, both excitation and inhibition coupling produce unconventional results. Carl and Abbott [77] study the converging results of two identical neurons described by three models (IF, general phase-coupled, and Hodgkin-Huxley) with non-instantaneous excitatory or inhibitory synapses between them. They find that in models with instantaneous (zero rise times) or extremely rapid synaptic responses, excitatory coupling leads to synchronization. However, if synaptic rise times are slower than the width of an action potential, inhibition rather than excitation produces synchrony. This work engendered subsequent research on the different delay effects on excitatory and inhibitory couplings. Ernst et al. [23,24] identify the mechanisms of synchronization of pulse-coupled oscillators when inducing temporal delays and inhibitory coupling. The simulation with $N >> 2$ oscillators reveals there exist multi-stable phase clustering with inhibitory coupling and spontaneous synchronization and desynchronization with excitatory coupling. Marc et al. [72,73] explain the perpetual synchronization and desynchronization switching under delayed excitatory

coupling is caused by the unstable attractors, which are enclosed by basins of attraction of other attractors but are remote from their own basin volume. Then, Ashwin and Marc [8] give the first rigorous demonstration of existence and robust occurrence of unstable attractors in the network of oscillators with delayed pulse coupling. Following this study, Henk et al. [11, 12] proves the unstable attractors are saddle periodic orbits whose stable sets have non-empty interiors. Additionally, there exist heteroclinic cycles between unstable attractors for a network of four oscillators for an open set of parameter values. In [9], Ashwin also discusses the function of this heteroclinic connections in the neural system information procession, weather patterns, spacecraft trajectories, etc. On the delayed inhibitory coupling, [71] shows that the synchronous state is stable and independent of the specific network topology. As the basic theoretic model of this thesis, the delayed inhibitory coupling will be discussed further in later chapters.

On the heterogeneity of oscillators, compare to the biological system (like firefly groups), it is relatively small and easy to compensate for computer hardware and software. Because the oscillating heterogeneity is not a topic of this thesis, it will not be discussed much here. On the other hand, the heterogeneity of network topology configuration plays a critical role on the synchronizing speed aspect [75]. Guardiola et al. [30] find the converging speed limit is close related to the network topology randomness degree. Using the the random matrix theory, Marc et al. [71] claim there is a synchronization speed limit determined by the typical in-degree per neuron to coordinate spiking activity. Qi et al. [59] study the speed limits of several kinds of complex networks (small-world, scale-free, and modular) with different eigenratios. Recently, the effect of small-world theory [82] and network complexity on the dynamics of pulse-coupled oscillator networks has also become a research hotspot [68].

2.3.3 Application of PCO in computer science

The mutual interacting characteristics and various converging results of the pulse-coupled oscillator network make it suitable for distribute network computing jobs. Except for the basic

synchronicity and desynchronicity primitive and their scheduled or asynchronous communication scheme, the partial clustering behavior of PCO imitated from the visual and brain neuron systems can be used for image segmentation and feature binding .

Synchronization

When using M-S model to provide time synchronization for wireless distributed networks, many realistic factors of the wireless environment have to be taken into account. To avoid the instantaneous coupling, reference [83] proposes that nodes accumulate the incoming pulses in the past period and do the phase adjustment once in all at the beginning point of the next period. While [74] enlarges the period from 1T to 2T to buffer the transmission delay. When the node gets a pulse at some point of the first T, it will react to the pulse at the same point of the second T. Our previous work [55] gets rid of the swing actions in the coupling procedure by predicting the final converging direction and improving the synchronizing speed and energy efficiency.

Desynchronization

Desynchronization, as a novel primitive, has received more notice because of the research on pulse-coupled oscillators' synchronicity function since 2007. Desynchronization implies that nodes perfectly interleave periodic events to occur in a round-robin schedule and can be useful in several fields [20]: medium access control in wireless transmission, high-throughput traffic intersections and even nodes burden distributing. The desynchronized results in PCO networks can be achieved by artificially coordinating the synchronized scale [52, 79], or directly converging from the special oscillator coupling mechanisms. In the research of PCO desynchronization, the oscillating and coupling function are further simplified into the phase adjusting function or Phase Response Curve (PRC). The response/adjusting is usually a function of current phase values of the oscillator itself or even its neighboring nodes. The most common function form are $sin()$ [4, 5, 48] and median - like which is used in most of the following literature. Ankit

and Julius [56] presents two biologically-inspired algorithms for achieving desynchronization: DESYNC and INVERSE-MS. Using DESYNC, Julius et al. [20] continues implement a self-maintaining algorithm DESYNC-TDMA to organize a collision-free TDMA schedule for transmitting wireless messages in wireless sensor network. Jonathan and Iain [70] present a lightweight adaptive Cyclic Duty Allocation Protocol upon the desynchronization primitive and evaluates its performance under both ideal and adverse network conditions. However, the algorithms and protocols presented above rely on the single-hop, all-to-all topology. References [19, 51] extend the desynchronization algorithms to multi-hop networks. They further inspect desynchronized configurations for several classes of topologies and discuss the relationship with other variants of graph coloring [40].

Data communication

In [65] a Phase Diffusion Time Division method (PDTD) is proposed, in which the independent phases of nodes are clustered, while those within the interaction range form the phase difference to appropriately avoid the collision. Thus the PDTD method can reduce the collision frequency while sustaining the sufficient throughput of communication. Extended from PDTD, A Phase Diffusion Synch-Alliance (PSDA) [66] is organized according to the node-type selection that is used to mark the repulsive force interactions between neighboring nodes, and self-adjusted according to the network topology. The PSDA provides coupled phase dynamics unified repulsive and synchronizing interactions to form an effective phase difference pattern to facilitate collision-free states in a fully distributed way. Wakamiya and Murata [79] propose a scheme for data fusion in sensor networks where information collected by sensors is periodically propagated without any centralized control from the edge of a sensor network to a base station using pulse-coupled synchronicity.

Image segmentation

De Liang Wang and D. Terman [80] propose a novel class of local excitatory, globally inhibitory oscillator networks (LEGION). The fired oscillator recruits the oscillators stimulated by the same pattern, while preventing others from jumping up. Simulation shows the network rapidly achieves both synchronization and desynchronization whether using relaxation or IF oscillator model. This model lays a physical foundation for the oscillatory correlation theory of feature binding and can be used on scene segmentation, figure/ground segregation, image analysis, etc. [14, 42, 44, 81]. The clustering algorithm SOON (Self-Organization of Oscillators Network) [60] associates one oscillator with each datum and defines both the excitatory and inhibitory interaction between oscillators according to the relative distance between the points. By incorporating concepts form relational and prototype-based clustering techniques, SOON can cluster high-dimensional data that is applicable in color image segmentation. In [62], SOON algorithm is implemented on more realistic biological neuron models [27] and produce positive results when applied to face image segmentation.

Hardware implementation

For the real hardware implementation, work [34] uses the narrow pulse characteristics of UWB systems to emulate the pulse-coupled integrate-and-fire (IF) model to achieve distributed synchronization. Bernard and Cesar [29] implement the LEGION model using FPGA (Field Programmable Gate Array) for image segmentation. Ravasz et al. [22] realize a system of electronic oscillators communicating through light-pulses. It is even useful in hardware project because it resolves wiring problems and global coupling can be achieved.

2.4 Summary

Under the supports of mathematical analysis and modeling work, the pulse-coupled oscillator system exhibits versatile abilities in computer science fields. The following chapters will focus on sync/desync, their clustering function, and applications in the network computing area.

Chapter 3

Structure and Theoretical Model

3.1 Structure

In this chapter, a complete structure of this thesis work on logical cluster is presented. The work has two parts: the General Criteria based Clustering method (GCC) and the data communication framework on the clustered networks. They are connected by the concept of logical cluster (LC). The relationships between these two parts, their applications, and service fields are all demonstrated.

As the construction method of Logical Cluster, GCC includes a clustering algorithm A based on the mathematics model M, a set of system parameters P, some physical technique requirements T and the result with property Q. The algorithm A is mathematically founded on the delayed inhibitory pulse-coupled model M. With the support of physical techniques T and parameter setting P, the algorithm A engineers this mathematical model into sensor networks by regarding sensor nodes as oscillators with the same oscillating function but different initial phases ϕ that are mapped from the selected property value. Two level concurrent operations among the nodes and processes in each node are executed to imitate the delayed inhibitory coupling activities. Iteratively, the system will self-converge into a designated number of Logical Clusters with

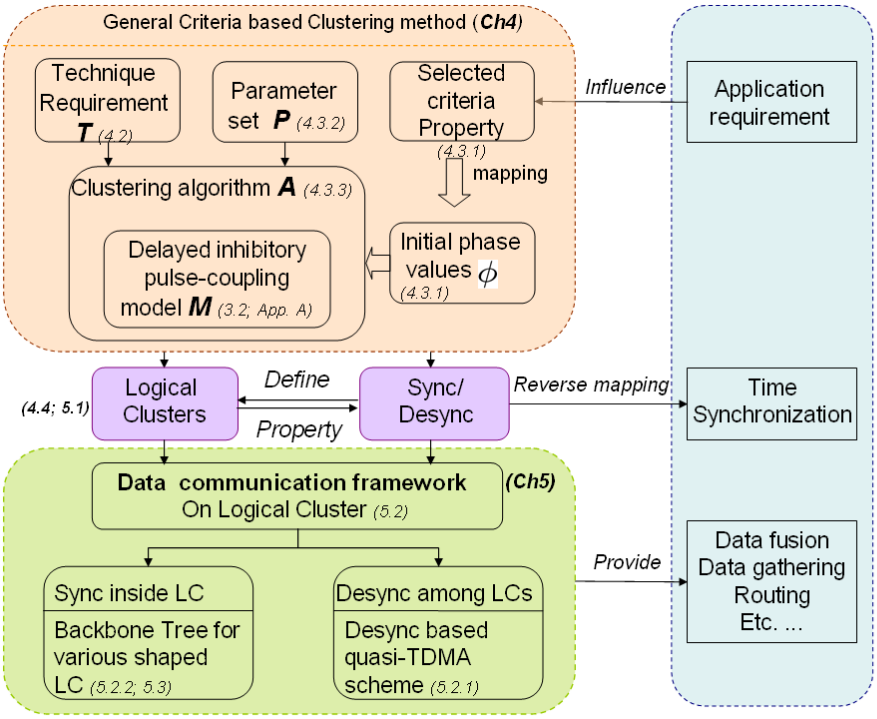


Figure 3.1: The relationships among clustering, data communication, and the application.

synchronicity and desynchronicity characteristics Q.

After the cluster construction, the uniqueness of logical clusters (represented as diverse physical shapes and sync/desync in and between LCs) requires a new communication scheme. Therefore, a data communication framework is proposed in Chapter 5. There are two components in the framework that solve the communication problem inside each LC and among them respectively. Inside the LC, a novel Backbone tree/forest topology is proposed to connect and route the data on the scattered cluster members. Among different LCs, a desynchronization supported quasi-TDMA scheme is constructed, which identifies the members and physical active areas of each logical cluster. Under the proper sleeping schedule, clusters report data to the sink or base station efficiently without interfering each other. With the cooperation of these two components, most of the data communication and transmission in the wireless sensor networks can be achieved.

In regards to the function domain of each part of the whole framework, the GCC method is completely independent and applicable for more extensive multi-node computing system (WSN is broadly regarded as one kind). The Backbone tree algorithm can also work independently in more general network situations. The quasi-TDMA component, however, is dependent on the desynchronization support from the preceding clustering method.

With the entire picture of construction, communication, and the logical cluster itself, every part can relate to the system or up-layer applications. In GCC, the user or application level requirement can influence the selection of the clustering criteria to make the clustering results become more beneficial for further procession and analysis. After successfully constructing logical clusters, through simple reverse mapping and calculation, the whole network can achieve local time synchronization and global desynchronization, which are prerequisite for many upper-level applications. Next, under the assistant of the communication framework, the logical clustered network can provide efficient data fusion, aggregation, routing, data gathering, and other services. Furthermore, under the support of these three basic primitives, additional and more concrete and further applications can be achieved.

3.2 Leaky Integrate-and-Fire Model

3.2.1 Integrate-and-Fire

Although there are many variants of oscillator models mentioned in Chapter 2, here the basic leaky integrate-and-fire (IF) model is used because it is simple and amenable to mathematical analysis. The 'IF' means that the potential x_i accumulates by integrating until up to some threshold x_{th}, then the oscillator fires and emits a pulse signal $\delta(t)$. After firing, the potential falls down to zero and starts to integrate again in the next cycle. The presentation of the following mathematical discussion follows the literature and adds additional clarity about the equation's characteristics.

The integrating speed of the potential is presented by the following differential equation:

$$\frac{dx_i(t)}{dt} = -x_i + I_0 + \sum_{j=1, j\neq i}^{N} J_{ij} P_j(t) \tag{3.1}$$

The change of potential is decided by the regular integrating speed and the instant leaps caused by the incoming pulses.

The first part of Equation (3.1),

$$\frac{dx_i(t)}{dt} = -x_i + I_0$$

shows that when the oscillator is isolated, the potential integrating speed is decreased as the potential value increases. Here $I_0 > x_{th}$ controls the natural period of the oscillator. $\frac{dx_i(t)}{dt} > 0$ and $d(\frac{dx_i(t)}{dt})/dt < 0$ mean the curve of the potential value is monotonically increasing and concave down.

When oscillator i is in a network and can receive fire signals from neighboring oscillators (j), its potential instant speed is adjusted by

$$\sum_{j=1, j\neq i}^{N} J_{ij} P_j(t)$$

Here J_{ij} is the coupling strength between nodes i and j.

$$P_j(t) = \sum_{m} \delta(t - \tau_j^{[m]})$$

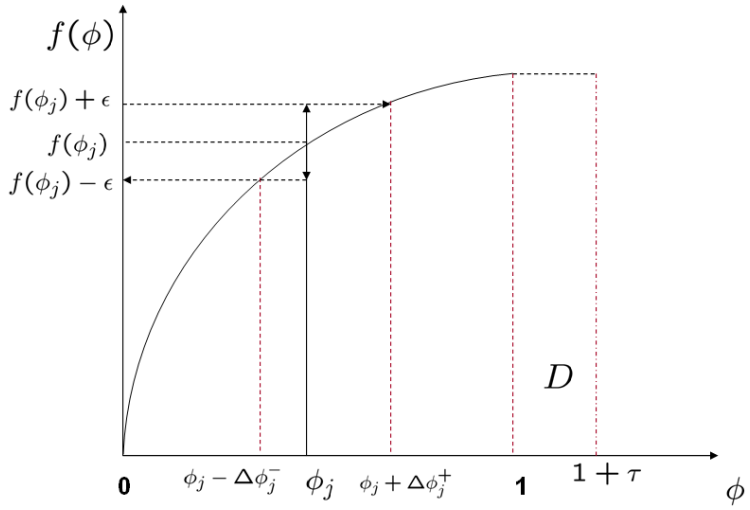


Figure 3.2: Potential $f(\phi)$ and phase ϕ adjustment during excitatory/inhibitory coupling.

represents all pulse signals from node j in the past, in which $\delta(t)$ is the Dirac delta pulse function and $\tau_j^{[m]}$ is the oscillator j's mth firing time of oscillator j.

Figure 3.2 shows the curve of an oscillator potential (Y axis) along with its phase (X axis) in an IF model. The potential is written as a function of phase ϕ:

$$x_i(t) = f(\phi_i(t)) \tag{3.2}$$

In which the phase variable $\phi_i(t) \in [0, 1]$ specifies the state of each oscillator i at time t and the potential function is simplified as f which is set to monotonically increase and concave down ($f' > 0$, $f'' < 0$), as in Equation (3.1). Normalized, set $x_{th} = 1$ so $f(\phi) \in [0, 1]$ and $f(0) = 0$, $f(1) = 1$. Because of the continuity and monotonicity of f in $[0, 1]$, the oscillator potential and its phase are affine one-by-one. The variable phase can reflect the characteristics of the oscillators potential. So in the rest of the paper we use phase ϕ as the main measurement to investigate the dynamics of the oscillators in networks.

3.2.2 Coupling scheme

When there is no coupling, the phase moves at the speed

$$\frac{d\phi_i}{dt} = \frac{1}{T} \tag{3.3}$$

in which T is the oscillating period. Whenever oscillator i reaches a threshold $\phi_i(t) = 1$, the phase is reset to zero $\phi_i(t^+) = 0$ and a pulse is sent to all oscillators, which receive this signal after a delay time τ. At time $t' = t + \tau_0$, the receivers react to the fire signal and adjust their phases according to:

$$\phi_j(t'^+) = \begin{cases} f^{-1}(f(\phi_j(t')) + \epsilon_t) = B, & \text{if } 0 < B < 1 \\ \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

in which

$$\epsilon_t = \frac{n_0(t)}{n}\epsilon$$

Because the phases are bounded in $[0, 1]$, if B's value is less than 0 or greater than 1, it would be regarded as 0. n is the network node number and $n_0(t)$ is the number of nodes which fire simultaneously at time t. As the converging degree increases, n_0 will increase from small value (1 or 2) to its maximum value, which is the number of nodes in each group under the totally converged state. ϵ is the coupling strength, factor $0 < \frac{n_0(t)}{n} < 1$ controls the amount of phase adjusting on each firing group. When $\epsilon > 0$, the potential will jump up by ϵ_t amount. When phase jumps forward a step, the integrating period will be shortened. The coupling is excitatory. When $\epsilon < 0$, the potential falls down, the phase jumps backwards and the period is elongated. The coupling is inhibitory (Figure 3.2).

The Equation (3.4) and Figure 3.2 suggest that starting from the same initial phase value, the different coupling schemes will cause different phase adjustment amounts and directions. The

Table 3.1: Different converging results of pulse-coupled Oscillator model

Concave down	Instant coupling	Excit	→ Sync [46, 50]
		Inhi	→ Async
	Delay coupling	Excit	→ Unstable clustered [8, 73]
		Inhi	→ Stable clustered [23, 24]
Concave up	Instant coupling	Excit	
		Inhi	→ Clustered and degrade [49]
	Delay coupling	Excit	
		Inhi	

resulted phase value will sequentially influence the system's next firing and coupling situation. Iteratively, the system will come out with entirely converging or non-converging final results. Without considering the refractory period and continuous firing reaction (avalanche effect), there are three main factors determining the coupling scheme: the concavity of oscillating function f, the sign of coupling strength ϵ and the existence of response delay τ. First of all, the non-linearity of the oscillating function is vial to the convergence of the PCO system. If the function is linear, the phase adjusting amounts of all oscillators will be the same no matter what their own phase values are. And the phase distances among them will never change, so the system will never converge or disseminate at all. If the oscillating function is concave down, under the same coupling strength, the bigger the phase value is, the larger the phase amount will be adjusted. While the concave up function causes the contrary changing situation. Second, the sign of coupling strength will advance or postpone the firing, which makes the phase attraction happen in a different position (0 or 1

point). While the existence of response delay complicates oscillators' converging and repulsive behavior, it also increases the system attractor number from a base of one. Table 3.1 lists the different results along with the different choices of these three elements. Given the inevitability of transmission delay in wireless environment and the stability of final states, we choose the concave down, delay considered inhibitory pulse-coupled scheme as the core mathematical model of this thesis.

3.2.3 Clustering in Delayed Inhibitory Pulse-Coupled oscillators

References [23] and [24] observed that when the coupling is inhibitive and always has τ delay after firing, the oscillators in the whole network will converge to several clusters. All members in one cluster share the same phase. Between clusters, each pair of the clusters have a fixed phase offset. They are phase-desynchronized (Figure 3.3). And this sync/desync state is stable because either the outside noises or minor phase deviations cannot break down the clustered state. Besides, empirical data (Figure 3.4) show that the number of converged clusters m is approximately inversely proportional to twice the length of delay τ:

$$m \approx 1/2\tau \tag{3.5}$$

Regarding the converging reason and why the clusters are of a size 2τ, the appendix A and reference [54] give out the a heuristical proof of two nodes' absorbtion and repulsion: whenever two oscillators' phase difference $|\phi_{ij}|$ is less than delay τ, after finite iterations, they will assuredly end up in a synchrony state; when $\tau < |\phi_{ij}| < 1 - \tau$, their phase lag will lock onto some fixed distance that equals $\phi_0 \in [\tau, 1 - \tau]$ and they will never synchronize, which is called the antiphase stable state. When there are more than two oscillators, Ernst et al. [24] explained that the system fixed point's attractive radius is around τ size. Thus "...the phase axis is partitioned into intervals of size $\approx 2\tau$, each of which being a basin of attraction for one cluster." Although numerous simulations confirmed this phenomenon, as far as I know, there is as of yet no strict mathematical

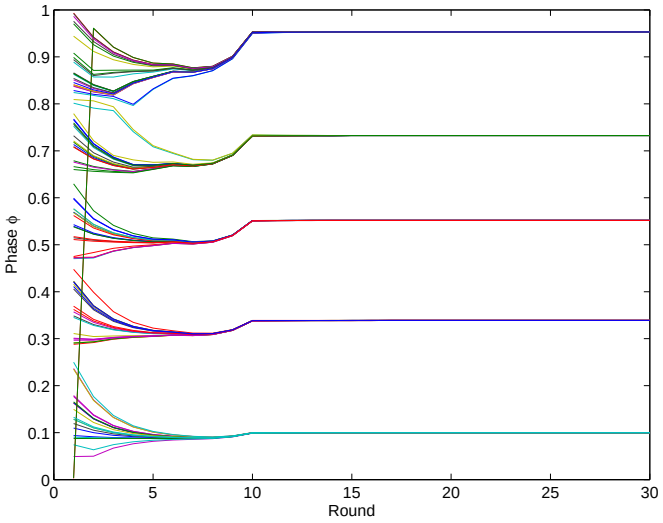


Figure 3.3: The converging procedure and clustered result under delayed inhibitive coupling($n = 100$, $\tau = 0.1$, $\epsilon = 0.2$) .

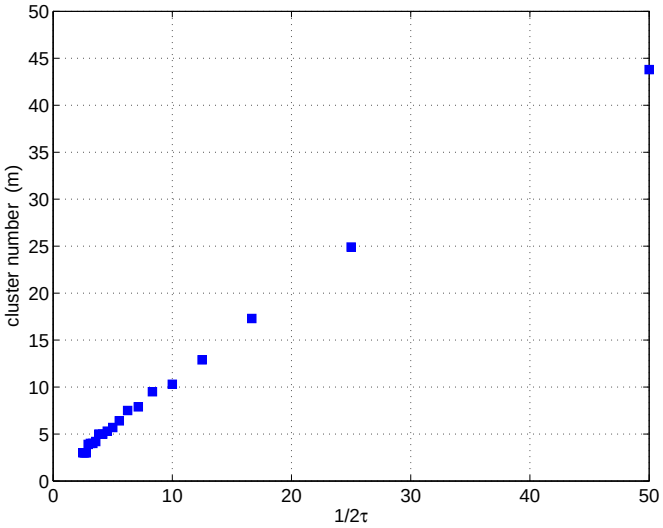


Figure 3.4: The proportional relationship between cluster number m and delay value τ.

proof or explanation on the convergence when the number of nodes is greater than 2, which will be one of our future works.

3.3 Summary

The basic mathematic model and theory basement is presented in this chapter. The stable converging results of the delayed inhibitory pulse-coupled model make possible the realization of logical clusters. Based on that, a complete structure of the construction and data communication framework is developed on logical clusters which will be discussed in detail in the next two chapters.

Chapter 4

General Criteria-based Clustering (GCC) method

4.1 Main Idea

The mathematical analysis and simulation results from last chapter show that the converging of the Delayed Inhibitory pulse-coupled Oscillator model (DIPCO) is the phase mutual synchronizing and desynchronizing procedure of oscillator population. The consequent sync/desyn results differentiate and partition the oscillators by the phase scale. The oscillators with the same final phase values default into abstract group, and different groups keep fixed phase difference (intervals).

In spite of the operation procedure, the input and output of DIPCO show that it is a one dimensional partition tool, which intercepts the phase value into several connected sections. Fixing all the other parameters, the partition results are totally determined by the initial phase distribution of the system. Here, the oscillators' initial phase values work as the abstract partition criterion.

Regarding each sensor node as a neuron oscillator and mapping some node data as its initial phase value, the mutual coupling can make them self-organize into a designated number of clus-

ters. Due to the abstractness of phase value, any numerical node data can be transferred as the oscillators phase values and work as the clustering criteria. Therefore, the method presented in this chapter is called the General Criteria based Clustering (GCC) method. As the cost of generality, this method does not specify any objective optimization performance of the clustering. On the contrary, GCC provide a basic operating framework and guidelines, in which the users, system or specific application can further select the clustering criteria and distance measures as required.

GCC accomplishes the networking node cluster result by the mutual synchronizing and desynchronizing operation. The method provide three very import primitives in sensor networks. Moreover, for the DIPCO, the simple calculation, simple pulse communicating form, finite iterating round and transmission delay concerning, all favor the method's suitability for the sensor network and hardware implementation.

This chapter presents how to engineer the DIPCO mathematical tool into wireless sensor networks to provide the three basic synchronization/desynchronization/clutering primitives. All the following steps and algorithms are presented from the individual node perspective for easy hardware implementing.

4.2 Technical Requirements

Before formally introducing the new clustering method, the physical requirements on system nodes and distributed networks are pointed out firstly.

(1) Each related node in the network is equipped with a hardware or software oscillator with fixed frequency. All oscillators should have the same oscillating frequency, but different initial phases are allowed.

(2) There is no limitation on the transmission media. The coupling signal can communicate in either wired or wireless networks. However, due to existing mature time synchronization protocols in wired networks together with the current emphasis on wireless networks, the remainder of this section is devoted to wireless technology.

(3) The firing signal used for causing coupling among nodes should be short, fast and low-cost. The only purpose of the signal is to inform other nodes about their firing time points, so the signals do not need to contain any source or target information. At the first stage of converging, the signal fires chaotically and frequently, so the firing signal should be fast and low-cost for energy consumption and channel interference considerations.

Several options are available. One option is based on Ultra Wide-Band (UWB) pulse radio [3, 35], which can be configured in a low bit rate, long-range transmission configuration [7, 10] to meet the requirements of the firing signal. A second option is based on Free Space Optical (FSO) pulse communication system, which supports long-distance point-to-point node communication and moreover can be configured to support broadcast [18]. In addition, since Radio Frequency (RF) systems are common, a more practical option is to use the short message supported by the existing transmission protocol, such as a 56-bits short-term synchronization head file in IEEE 802.11g [1]. The detection of a pulse message is dependent on which technique is deployed: for example, in optical pulse systems the detector (e.g. photodiode) can be used to detect multiple pulses [28, 78]; for UWB, the optimum detector is simply the energy detector with an appropriate set threshold value [34].

(4) All nodes are located in a single cell. To guarantee the convergence of the oscillators to the clusters, the system requires that the coupling is of the all-to-all form, which means that when one node fires and emits a signal, all nodes in the network should receive it and react to it. Simple broadcast in a single cell is sufficient for all-to-all coupling. When the network scale is beyond one hop, relay broadcasting of the firing signal is required at the periphery of the cell. This relay will increase the transmission delay. The more relay times, the lager indetermination there will be on delay value, which is caused by interference, message lost, etc. A key requirement for the stable clustering result of DIPCO is that all nodes should have the same length of delay. The increase of indeterminate delay will make the setting and adjusting of the delay parameter more complex. As this is beyond the scope of this paper, only the single cell case will be discussed.

Although this paper contains examples and simulations given in the wireless sensor network environment, the GCC method is not only limited to it. This method is applicable to all the multi-node computing systems that meet these requirements.

4.3 Method Steps

The GCC method has two major procedures: Mapping and Converging, which are further divided into four steps. Mapping is used to transfer the property data and values in the physical system to the variables and parameters of the mathematic system. It includes the two steps of criteria selection and parameter setting. The converging procedure includes the two steps of coupling and complementing. By imitating the coupling activity of neuron oscillators, the computing nodes can self-converge into a designated number of logical clusters with sync and desync characteristics. The complementary step provides period elongation, as well as reverse mapping two optional operations for further system and application requiring.

4.3.1 Step 1. Clustering criteria selection and mapping

Like mentioned in Section 4.1, using DIPCO as an one dimensional partition tool, the system initial phase distribution determines the final partition result. Any node, system-related data or system-related properties can be mapped into the phase domain as oscillators' initial phase values through some transformation. Thus, both selecting the kind of node data as the partition criterion and mapping the physical attributes of sensor nodes into the mathematical phase domain are two problems upfront.

Based on the iterative operating procedure and the final converged stable equilibrium state of DIPCO, there should have been one or a few items in phase domain that were optimized (maximized, minimized, or most balanced). Here are some observations from the final equilibrium state of the DIPCO.

1) The oscillator firing time point has tansformed from total randomness to some orderliness,

in which the firing timing and strength of groups is recurrent and predictable; the number of group absorption (group number decrease) caused by the phase synchronizing is zero.

2) $\phi_{(x,x-1)}$ represents the phase distance between group G_x and the next firing group G_{x-1}. Assuming group G_x always fire just before G_{x-1}, then $\Delta\phi_{(x,x-1),n_x}$ means the distance changing amount caused by reacting the pulse from group G_x that has n_x members. This enlarged amount of phase distance caused by reacting the firing with the weight n_x once should be equal to the sum of compressed amounts of phase distances by reacting the multiple fires from all the other $m-1$ groups in which m is the group number in the stable state.

$$\Delta\phi_{(x,x-1),n_x} = -\sum_{i=1,i\neq x}^{i=m} \Delta\phi_{(x,x-1),n_i}$$

Because of this equality, after one round coupling, the intervals between any pair of oscillators maintains the same length.

3) The minimum value of a phase distance between two groups after $m-1$ times compression in one round should remain larger than the system delay value τ, otherwise the two groups will agglomerate into one.

$$MIN(\phi'_{(x,x-1)}) > \tau$$

4) The total compression amount of one phase interval (distance) $\phi_{(x,x-1)}$ caused by $m-1$ times coupling in one round is equal to the sum of the compression amounts of all the other $m-1$ intervals caused by one coupling with the group G_x.

$$\sum_{i=1}^{i=m-1} \Delta\phi_{(x-i,x-i-1),n_x} = \sum_{i=1,i\neq x}^{i=m} \Delta\phi_{(x,x-1),n_i}$$

5) The action power of any group's weight (the member number) and its following interval length is equal with each other.

$$P_i(n_i, \phi_{(i,i-1)}) = P_j(n_j, \phi_{(j,j-1)})$$

with $i, j \in (1, 2, 3, ...m)$, $i \neq j$.

From the theoretic analysis and above observations, the minimized item could be the entropy (orderliness) of the system, or some cost function. However, because the converging mechanism behind these optimizing function is still a missing puzzle, it is hard to connect the quantity objective in a real physical system and the associated physical parameter to the theoretical equilibrium state and its control mechanism and factors. Under the limitation of currently being unable to specifically design some optimization function of clustering, instead the GCC provides the most basic function of node clustering, which works as a network organizing method and aims at increasing scalability, simplifying network management, and saving energy. In term of higher level optimization some network performance (such as minimizing energy consumption, maximizing longevity, deciding the most appropriate parameter values, etc.) is not specifically designed in GCC.

After clarifying GCC's clustering performance level, the below analysis provides some possible partition criteria candidates and the rules to select the criteria.

Each sensor node, as the unit with certain computing/transmitting abilities and memories, is loaded with all kinds of node properties, content data, soft/hardware parameters, etc. Among all these candidate properties, as long as they are numerable and the values are bounded in some range, they can work as the clustering criteria.

The selection of proper criterion property is dependent on the purpose of the cluster formation, which is often associated with the application. Multiple criteria may be chosen for different reasons thereby enabling different possible cluster reformation. The GCC builds a connection between application requirements and clustering construction due to the criteria selection. Such a connection means that a known objective may lead to criterion selection, or a criterion can be firstly selected that results in some benefit. For example in WSN, for the objective of network

work load balancing, the node's residual energy amount can be used as the clustering criterion. Nodes with similar residual energy level are thus grouped, and according to the different energy levels, the cluster members can adjust the sampling and reporting rate so as to balance the load and extend the network lifetime. Or conversely, if using node sensed content as the clustering criteria, then it may be easier to perform data aggregation inside clusters whose members have similar content data. The final condensed information would be shorter and consume less transmission energy.

All the qualified node data or properties form a criteria candidate set $C = \{C_1, C_2, C_3, ...\}$. From the set, one or several candidates may be appropriately chosen by a user or autonomously in an automatic application. Based on the selected single or multiple criterion prosperities, there is a need to set a distance function before entering the phase domain. Reasons for this step are twofold. Firstly, the initial phase is a one-dimensional scalar variable in $[0, 1]$. For the higher-dimensional data (like X,Y-coordinates) or selected multiple criteria items, the distance function could decrease the data dimension. Although there is some mathematical work on multi-dimension phase convergence [52], it is still too complicated to be used in a practical system at this time. The other reason is that the definition of a distance function itself has specific physical means that could be the Euclidean distance between two physical points or some similarity measure distance. The different definitions of the distance function will influence different characteristics of the partition results. Some simple examples are given here.

Define the distance d_i as single scalar variable (like node id, sensed temperature, residual energy) or simple combination of several variables with different weights, then

$$d_i = \sum_{x \in C} w_x c_{xi} \tag{4.1}$$

Define the physical distance from current node (x_i, y_i) to some fixed point (x_0, y_0) (which could be the location of the sink node, base station or event happened point) as the distance measure, then

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \tag{4.2}$$

Define the distance as the angle to the baseline of polar coordinates determined by point (x_0, y_0) and line $y = kx + b$, then

$$d_i = \theta_i = \arctan \frac{y_i/x_i - k}{1 + k \times (y_i/x_i)} \tag{4.3}$$

With different distance function defining, each node should be aware of the possible boundary value $[d_{min}, d_{max}]$ of this distance measure, which is calculated based on knowledge prior to the deployment or obtained from the broadcast message before clustering starts. With this information, the initial phase values can be mapped from the distance value by normalizing:

$$\phi_i = \frac{d_i - d_{min}}{d_{max} - d_{min}} \in (0, 1) \tag{4.4}$$

4.3.2 Step 2. Parameter setting

After choosing the appropriate clustering criteria, the next step is to set parameters in both the physical and mathematical systems.

After obtaining the initial phases, in order to calculate the consequent phase values at some specific time, the moving speed of the phase is needed. As described in Equation (3.3), the ordinary moving speed of the phase without coupling is decided by the oscillating period T. The oscillating period T takes the same value T_0 in both physical system and the mathematical model. The value of T_0 can be set rather freely under the constraint of the system delay and the expected cluster number m, which is discussed in the following paragraphs. Similarly, the length of T_0 will affect the whole system's converging time and corresponding energy consumption. The latter Step 4 will discuss this influence.

Delays are inevitable in most real networks, especially in wireless networks. The delayed inhibitory coupling model considers the transmission delay directly, in which after some nodes

fire, instead of reacting instantaneously, receivers wait for some time and then react. In real networks, the transmission delay mainly includes: transmitting delay t_{tx} in the sender, propagation delay t_{pg} in the air and decoding delay t_{dec} in the receiver [74]. Compared to the t_{tx} and t_{dec}, the propagation time is negligible. On the other hand, t_{dec} is measured by the receiver itself and thus is known. The only unknown item is t_{tx}. If the exact value of t_{tx} for every signal is needed, then the exact timestamps can piggyback on the signal message like in Reference [39]. However, in the GCC method, all the fire messages are the same short, pulse-like signals. Their emitting and receiving can be done by hardware or firmware using almost the same time. If the precision requirement is not very high, for simplicity, the transmission delay of all nodes are considered the same. $t_0 = t_{tx} + t_{dec}$ represents the total delay for a firing signal between its emission and reception. t_0 is the minimum allowed delay length and the real delay value t_τ should not be shorter than t_0: $t_\tau \geq t_0$. Normalize the time delay for the system model:

$$\tau = \frac{t_\tau}{T_0} \tag{4.5}$$

Considering $\tau \in [0, 0.5]$ and the relation in Equation (3.5), the delay value also sets the floor boundary of the real oscillating period T_0.

$$T_0 \approx 2mt_\tau \geq 2mt_0 \tag{4.6}$$

The coupling strength coefficient ϵ measures the inhibitory phase adjustment amount when receiving a firing signal. The desynchronized grouping phenomena and the relationship of Equation (3.5) are only tenable below some coupling strength threshold. The coupling strength is also critical for the converging speed and final cluster number veracity. We will discuss the empirical choice of the value of coupling strength in the simulation Section 4.5.

Table 4.1 shows the initial phase mapping and the relationship between the physical parameters and the system parameters.

There are two types of parameters in the clustering method: control parameter and normal system parameter. The control parameter could both qualitatively determine the converging result

Table 4.1: Mapping from Physical to System parameter

Physical parameter	**System parameter**
Criteria property $[min, max]$	Phase ϕ $[0, 1]$
Oscillating period T_0	T_0, $\frac{d\phi}{dt} = \frac{1}{T_0}$
$t_{delay} = t_{tx} + t_{pg} + t_{dec} \approx t_{tx} + t_{dec} = t_0, t_\tau \geq t_0$	$\tau = \frac{t_\tau}{T_0} \in [0, 0.5]$
Cluster number m	$m \approx \frac{1}{2\tau}$
	Coupling strength ϵ

and quantitatively influence system performance. For examples, the monotonicity and concavity of oscillating function f, the signs of coupling strength ϵ and the zero or nonzero of delay τ value jointly determine the node coupling form and the converging results. Meanwhile, the extent to f's concavity and the absolute value of ϵ both could influence the converging speed of systems, and the length of delay τ could roughly determine the final cluster numbers. The normal system parameters are usually used to describe the problem scale or the system results, such as the network node numbers n, final cluster number m, converging iteration round l, etc. Some of them could also influence the performance of the system, such as the length of the oscillating period T_0. Both the control parameters and normal system parameters are included in the parameter set $P = \{f, \epsilon, \tau, T_0, n, m, l\}$.

4.3.3 Step 3. Clustering

We regard every sensor node as an oscillator and map the selected criterion property as the oscillator's initial phase value. Based on theoretical results and analysis in Section 3.2, if the oscillators' potential and phase variables follow the preset oscillating function and execute the inhibitive coupling with the same delay length, after finite iterations, the nodes will be grouped into several clusters by their self-converged phase values (Figure 3.3). The following presents the clustering algorithm operated on each sensor node.

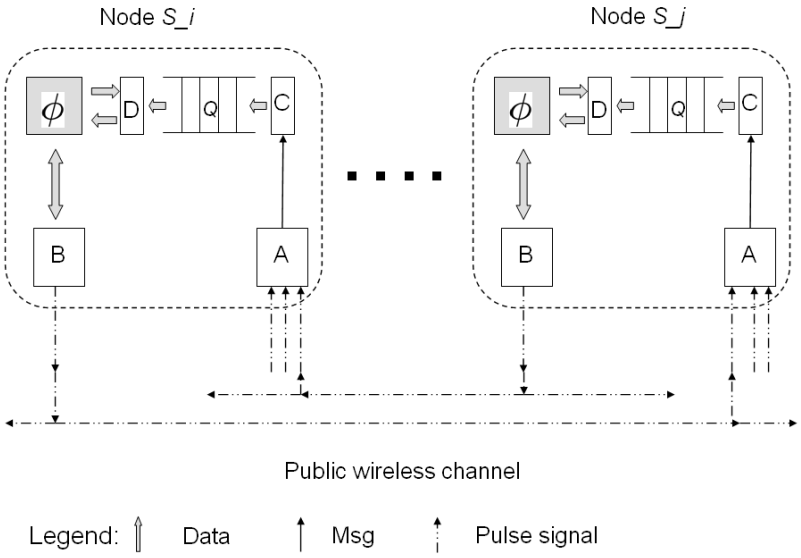


Figure 4.1: Two level concurrency among nodes and inside each node.

During the whole clustering procedure, each node mainly executes two operations: periodical self-oscillating and coupling with other nodes. The couplings among the nodes are mutually triggered by the same anonymous pulse-like signals, so each node also operates two assistant signal-receiving and broadcasting processes. Whether self-oscillating or coupling, the operation results are embodied on the node's phase value ϕ. There are four processes concurrently executed on each node: A, channel listening; B, periodical phase value self-increasing and resetting; C, signal receiving; and D, coupling-caused phase value adjustment. Processes A and C are asynchronous communication processes; B and D perform concurrent read/write computations on the shared node phase ϕ, that is, the ϕ is stored in a shared memory that is accessible by these processes. In addition, due to the delayed coupling model, the node uses a queue structure Q to store the waiting signals between processes C and D (Figure 4.1). The node's local clock time is used by all the processes. Figure 4.1 illustrates the four processes working function inside each node, and Algorithm 1 gives specific details about the required calculations.

Algorithm 1 Logical Clustering()

Require: Each node have the same f, ϵ, τ, T_0, t_0 value and different initial phase value ϕ_i

Ensure: System stable clustered

1. **repeat**
2. (**Process A**) Listen to the public wireless channel
3. (**Process B**) Increase the phase value ϕ at the speed of: $\frac{d\phi}{dt} = \frac{1}{T_0}$
4. **if** ($\phi = 1$) **then**
5. Broadcast $Firing\_signal()$
6. Reset $\phi = 0$
7. **end if**
8. (**Process C**)
9. **if** (Heard firing signals) **then**
10. Record current local clock time t_k and calculate the expected reacting time $t_{rk} = t_k + (\tau \times T_0 - t_0)$
11. Estimate the firing number $n_0(t_k)$
12. Add t_{rk} and $n_0(t_k)$ into the waiting *Queue*
13. **end if**
14. (**Process D**)
15. **if** (The *Queue* is not empty) **then**
16. Read the first item in the *Queue*
17. Obtain the upcoming reacting time $t_{rk'}$ and associated $n_0(t_{k'})$
18. **if** ($t = t_{rk'}$) **then**
19. Read the current phase value $\phi(t)$
20. Adjust the phase value using Equation (3.4) with $\phi(t)$ and $n_0(t_{k'})$
21. **end if**
22. **end if**
23. **until** Detect fixed number and strength of fire signals sequence in continuous periods

When not performing the coupling, process B in real-time updates ϕ at the speed of $\frac{d\phi}{dt} = \frac{1}{T_0}$, which is an instance of Equation (3.3). When process B observes that phase value arrives at the threshold value of $\phi \geq 1$, it broadcasts a firing signal to the whole network, while also updating the phase as zero $\phi = 0$. Process A continues listening to the public wireless channel and detects if there are any firing signals, and its detected results are only reported to process C. When the node detects firing signals, process C is activated. It records the current node local clock time t_k and estimates the firing numbers of the signal $n_0(t_k)$. The expected reacting time $t_{rk} = t_k + (\tau \times T_0 - t_0)$ and the firing number $n_0(t_k)$ are added to the end of the reacting queue Q of this node. After that, process C is suspended until the next firing signal is reported. Once the queue is not empty, process D reads the first item in Q and obtains the next expected reacting time $t_{rk'}$ and associated $n_0(t_{k'})$. Process D waits until the local clock time $t = t_{rk'}$, then it reads the current phase value $\phi(t)$; using Equation (3.4) and $n_0(t_{k'})$, it calculates the new phase value $\phi(t^+)$ after the coupling and updates this new value into the shared memory ϕ storage. Termination of all processes occurs when a sequence of firing signals with fixed number and fixed strength are repeatedly reported to process D (via the queue) for continuous several clock periods.

Both processes B and D can access the ϕ memory exclusively, which means that when process B is increasing or resetting the ϕ, process D cannot do the adjustment on ϕ, and vice versa. The read&write in both processes (Algorithm 1, lines 3, 6 for process B; lines 19-20 for process D) should be transaction operations. Because the speed of phase self-increasing is quick (simple adding or resetting), process B's memory holding does not significantly influence process D. However, process D's processing time is relatively longer than B's; here we set that the phase self-increase updating interval of B is always longer than D's accessing time for ϕ. Given this criterion as valid, the whole concurrency works. Due to the fact that phase self-increasing and resetting are regular actions for the node, under the same circumstances, process D always has higher accessing priority than process B: specifically, $Coupling\_adjust > Normal\_shift > Reset$.

All the control parameters (initial phase value ϕ, system delay τ, oscillating function f, period

T, coupling strength epsilon ϵ, etc.) is set before the coupling. Once the coupling starts, only the nodes periodical phase self-increasing and the incoming firing pulse causing abrupt phase adjustment are allowed. The outside parameters will not interference with the system's self-converging; the phase values of oscillators are mutually attractive or repulsive, which are decided by the fix points (attractors) and their attractive radius of the dynamic system. Finally the system will self-converge into a stable state in which the attractive and repulsive trends agree to the equilibrium state and it is immune to weak noise and disturbing. After that, comes whether the self-shifting and incoming pulse will not change the system cluster configuration. The whole converging procedure requires several oscillating rounds; the process in each node automatically judge the end of clustering by detecting if there appear a fixed number and strength of fire signals sequence in continuous periods.

From Algorithm 1 and Figure 4.1 we can see there consists of two levels of concurrency: the upper-level derived from a neuron dynamic system consists of all of the sensor nodes in the environment such that the nodes self-autonomously cooperate via pulse-coupled communications to effect a global clustering; the lower-level is specific to each node and consists of asynchronous node-to-node communication support, as well as asynchronous and shared memory internal communication. All the nodes operate individually and there is no central control message or data flow. The algorithm is distributed and the whole system is totally autonomic.

4.3.4 Step 4. Period elongation and reverse mapping

After the system converging, each cluster acts as a whole and m clusters fire alternately in one period. The firing pulse number decreases substantially. The superposed pulses from simultaneous firing cluster members are used to maintain the clustering topology of the network and keep it stable. Because of the independence of the clustering method, the cluster head election or rotation inside will not influence the macro network cluster distribution. Without special requirements, the network re-clustering period could be very long.

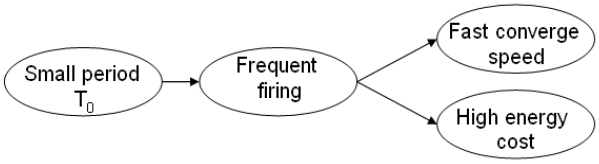


Figure 4.2: The influence of period to the converging speed and energy consumption.

During the procedures of converging, a short oscillating period is preferred. With the shorter period, the coupling happens more frequently and the system converges quicker. But after the converging becomes stable, if the system still keeps the short period, the emitting of maintenance pulse will be too frequent (Figure 4.2). Besides, the sensor nodes also need sufficient time to fulfill their other duties between fires. Therefore, the ideal situation is to use short T_0 during the converging procedure, increase the converging speed, decrease the time costs, and use an appropriate longer T_0 after converging becomes stable, firing occasionally to maintain the clusters and thereby decreasing maintenance costs. This method needs to elongate the period of T_0 after converging smoothly without disturbing the stable network state. To obtain this, there are two points to note. First, the new period should be integer multiples of the old period, i.e. $2T_0$, $5T_0$,..., etc. Second, the system precision should be high enough so as not to split the already converged clusters. This period enlarged function will be tested in the next simulation part.

As mentioned in Chapter 1, the members are phase-synchronized inside a cluster. If the application requires time synchronization among nodes, the phases can be reversely mapped to some accumulated time variables. Because the member nodes' phase and frequency are all the same, the transformed timescale must be the same as well, so all these sensor nodes are time-synchronized.

4.4 Logical Cluster

The clusters are divided based on the initial phase that is also the selected criteria properties of sensor nodes. The produced cluster members share the similar criterion property values, but their

physical location may or may not be near as those in traditional clusters. The cluster members are logically related and so the formed clusters are called Logical Clusters (LCs). Because the clustering is totally self-organized, there are no cluster head nodes either before or during the clustering procedure. The logical clusters' physical shapes are also more diverse than the normal ones from the existing algorithms. The following Table 4.2 lists the quality comparison between Logical Cluster from GCC and normal physical cluster. The more extensive quantity comparison will also be explained in the following section.

4.5 Numerical Results and Analysis

The simulation is divided into three parts. First the feasibility of GCC method is checked by demonstrating the produced novel logical clusters. Then the cost of GCC method is discussed. Finally, the stability of the clustering results under an enlarged period is verified.

The basic simulation environment is set to meet the requirements in Section 4.2. Three hundred nodes are randomly deployed in a 100×100 meter size field. Each node uses the uniform oscillating function $f(\phi) = \frac{1}{b} ln[(e^b - 1)\phi + 1]$ [50], in which $b = 3$ measures the extent to which f is concave down. Here we assume the transmission media is wireless. The firing signal is chosen to be the 802.11g short preamble - a specific sequence of pulses that has the length of 56-bits. The field is set as a single cell where firing signals are directly broadcasted and received and no relay work is used. If needed, there will be other environmental parameters given in each part.

4.5.1 Logical Cluster illustrations

To check the generality of the GCC criteria, here three kinds of representative criteria are chosen as examples.

(1) The location data of each node.

In Figure 4.3, the geography data criteria is the angle to the base line of a polar coordinate (defined by point $(20, 40)$ and line $y = 40$), in which Equation (4.3) works as the distance func-

Table 4.2: Concept comparison of logical and physical cluster

	Logical Cluster	**Physical Cluster**
Construction idea	Dividing nodes into groups by sync and desync Select criteria properties and mapping them to phase domain	Finding optimized cluster number and cluster head location Nodes join the nearest or minimum cost head realm
Characteristics	No cluster head designated beforehand Preset cluster number Globalized coupling	Specifically decide cluster heads Preset or self-organized Localized recruiting
Correlation among members	Close in criterion properties values Phase synchronized *Logically related*	Close in physical distance to head *Physically closed*
Cluster shape	Continuous or discrete Various shapes	Continuous area Polygonal/round shape
Inter communication	Desync supported Quasi-TDMA schedule Assign private time slot to each cluster self-maintainable	Different coding (CDMA, DSSS) Multi-hop routing or hierarchical multi-layer
Intra communication	Any topology (Backbone tree) Head election and rotation Synchronization baseline	Head centric star topology Predesignated, seldom rotate CSMA or head organized TDMA

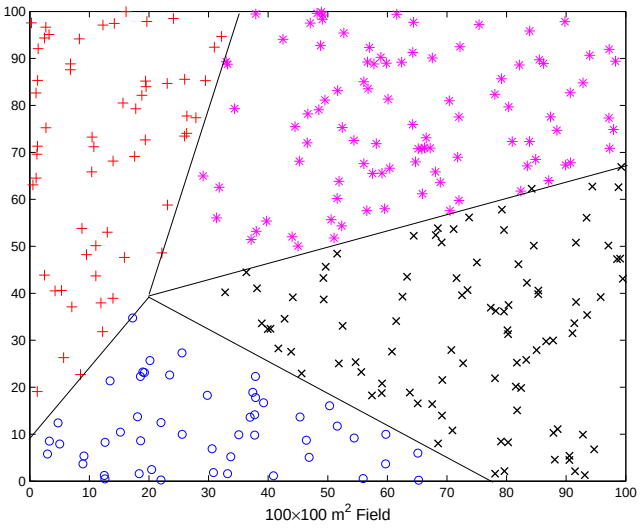


Figure 4.3: Normal polygon-shaped clusters based on angle to (20,40) and y=40.

tion. For comparison, the location based physical clusters from protocol HEED (Hybrid, Energy-Efficient, Distributed Clustering) approach [87] and LEACH (Low Energy Adaptive Clustering Hierarchy) are drawn in Figure 4.4 and Figure 4.5 respectively. The example shows that the GCC method is compatible with the traditional clustering algorithms that the polygonal logical clusters are similar to the normal ones, except there is no pre-designated head nodes. Figures 4.6 and 4.7 select the distance to the field point $(50, 50)$ and the outside field line $y = 1/2x + 200$ respectively as the criteria, and then proceed to map the distance values as their initial phases. The formed cluster shapes are circles and strips respectively. As discussed in Step 1, the physical borderline of each cluster is clear and these kinds of clusters are suitable for hotspot surveillance applications.

(2) Content-based clustering criteria.

In a $702.5 \times 310 \times 470$ cm^3 size room, 80 observing points, which form a $4 \times 4 \times 5$ three-dimensional grid of sensors, are set to measure the indoor air temperature of different places. At the right corner of room ($(690, 5, 275)$, star mark in Figure 4.8), there stands an air conditioner blowing out cold air. The nearer to the air conditioner, the lower the air temperature is. The temperature values are determined from a simulation using a simple linear distribution. The tem-

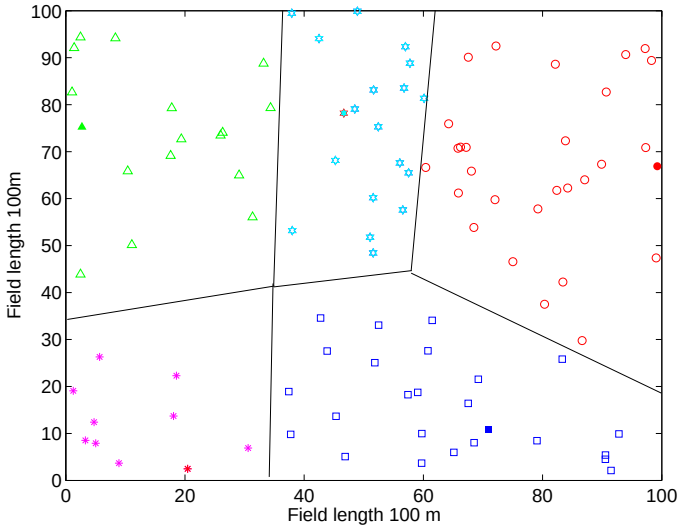


Figure 4.4: Polygonal cluster result from HEED (N=100, r=40).

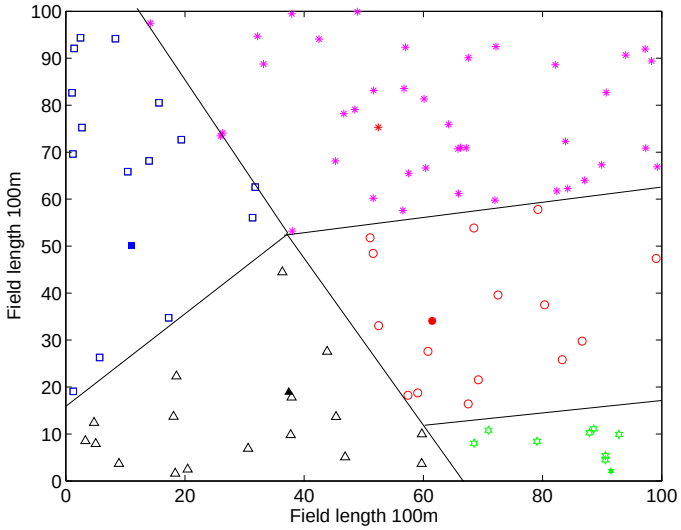


Figure 4.5: Polygon cluster result from LEACH (N=100, 5% nodes work as Cluster Heads).

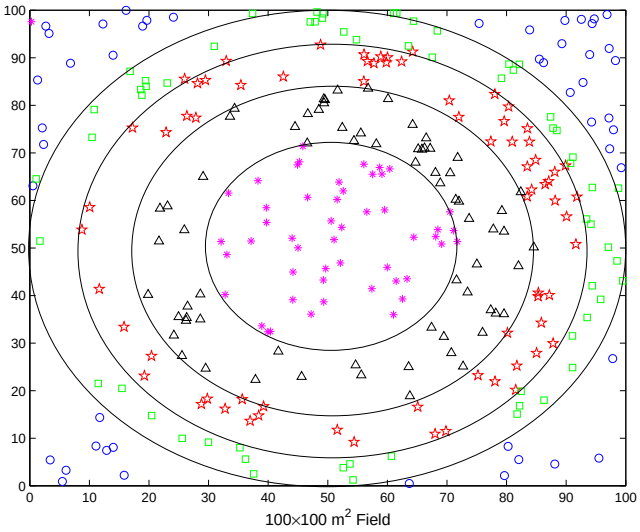


Figure 4.6: Concentric clusters based on distance to point (50,50).

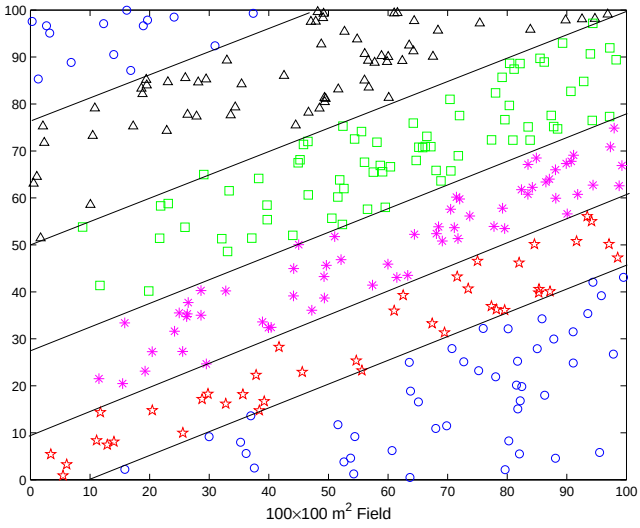


Figure 4.7: Strip clusters based on distance to line y=1/2x+200.

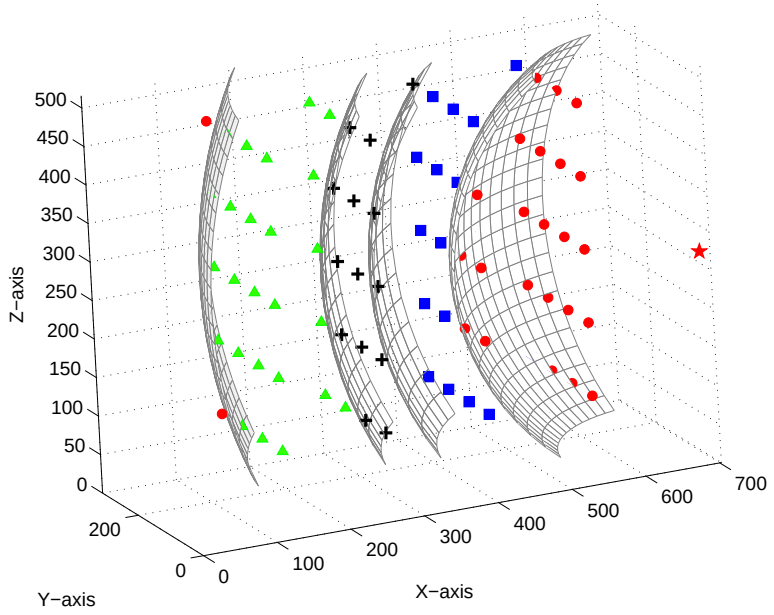


Figure 4.8: Three-dimensional clusters based on indoor temperature.

perature values are mapped to the initial phase and system parameters ($\epsilon = 0.3$, $\tau = 0.1$) are set as discussed in Section 4.3 steps. The 80 nodes are converged into 4 clusters. Figure 4.8 shows the three-dimensional clusters based on the node-sensed content, in which the contour surfaces are shown to highlight the clusters.

(3) The unique node ID.

This selection procedures scatter clusters as shown in Figure 4.9. The scatter cluster shows the key idea of the logical cluster. The clusters are overlapped in a physical field and cluster members are scattered everywhere. Intuitionally, these clusters have no physical meanings, but the cluster members are logically related and share some common points. This concept is similar with the online social network. The virtual social group share the same interests and based on that new ideas may emerge.

From Figures 4.7 and 4.8 we notice that some of the highest phase nodes and lowest phase nodes are clustered into one group. This circumstance occurs because the node phase is a modular

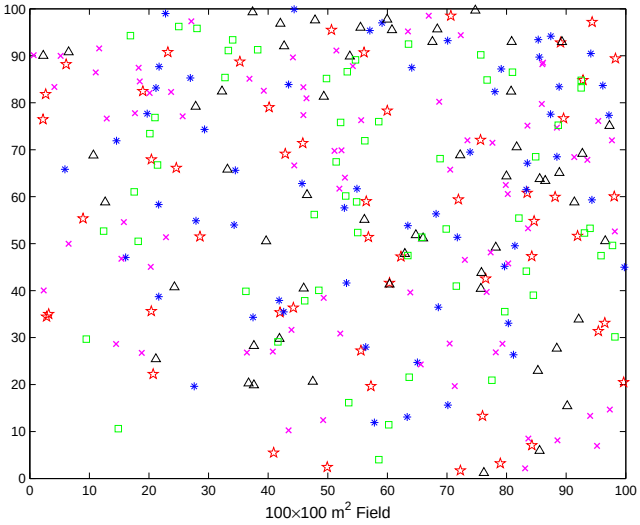


Figure 4.9: Scattered logical clusters based on node ID.

variable in the oscillator model: the phase value near 1 also means near 0. Given the phases of nodes whose criterion property values near minimum or maximum are close, there exists the possibility of clustering into one group. This outcome will not be a problem when the logical cluster is also modular (like the example of life-surviving temperature clustering - both the highest and lowest temperatures are not suitable for life to survive - clustering them into one group is reasonable). However, in other situations, these two contradictory classes may cause confusion. They need to be subdivided inside the cluster. In some cases, with the help of physical base parameters, it would be easy to differentiate the two class nodes (like in Figure 4.7, the physical location is substantially different between the two class nodes). A more general manner relies on each node's firing times. Based on the property of the oscillating function f and the phase coupling Formula (3.4), the firing order of nodes never changes during the system converging procedure. Thus, although the two classes of nodes may converge and they fire together, the higher phase nodes always fire one time more than the lower nodes due to the former group's different initial phase values. By comparing their firing times, the nodes can subdivide themselves.

4.5.2 Converging speed and complexity

The previous discussion focused on the possible cluster results. Now the clustering costs are given in greater detail. Here cost refers to the time and energy consumed during the procedures from random initial distributed network status to the completely clustered state.

The time for all the nodes in a network to fire once is defined as one convergence iteration (I), and it is an important metric to measure the algorithm's time and computing complexity. The length of one iteration T_I is a little longer than one oscillating period T because of the backward jump of phases in the inhibitive coupling. For the all-to-all coupling scheme, every node fires once in one round and reacts to the firing signal at most n times. This pattern occurs because the signals that emit simultaneously are only received and reacted once. Assuming the system converged in I_0 iterations, then it costs around $I_0 \times T_I$ time to accomplish the clustering, and during that period every node on average fires I_0 times and reacts less than $I_0 \times n$ times.

4.5.3 Performance impact factors

There are several factors that may influence the converging speed: the initial phase distribution, the coupling strength ϵ, the extent of f's concavity b and the node number n. The initial phase distribution is determined by the clustering criteria. f's concavity b has the same effect with coupling strength ϵ - both of which influence the phase jump amplitude in the coupling. Figure 4.10 shows the relationship between converging iteration number I and coupling strength ϵ under the different node numbers n. In Figure 4.10, the four lines take the randomly uniform initial phase distribution, and their resulting data are averaged over 500 times initial phase realizations. As predicted, the stronger the coupling strength is, the faster the system converges. The node number has little influence on converging speed because the lines with different node numbers almost superpose. Therefore, we assume that the converging iteration number I is independent to the system node number n, and the computing complexity of GCC is $O(n)$.

As is shown in Figure 4.10, when the coupling strength is strong, it costs fewer rounds and

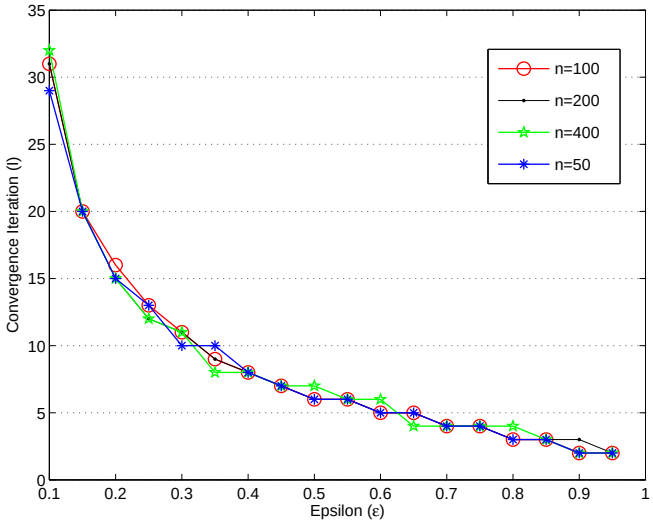


Figure 4.10: Converging round with respect to coupling strength under different node number.

associated energy for each node to achieve convergence. However, too strong on an adjustment in each step may combine the neighboring clusters that should have been separated. This situation engenders the control precision lapses of the expected cluster number. Therefore, the value of ϵ is critical to converging speed, energy consumption and the cluster number veracity.

Next we will show how to decide a proper coupling strength ϵ in a network to meet the high veracity of the clustering number with possible low energy consumption at the same time. To calculate the energy consumption, the first order radio model [31, 32] is used. Each sensor node will consume the following E_{Tx} amount of energy to transmit an l-bits message over distance d:

$$\begin{aligned} E_{Tx}(l,d) &= E_{Tx-elec}(l) + E_{Tx-amp}(l,d) \\ &= \begin{cases} lE_{elec} + l\epsilon_{fs}d^2, & d < d_0 \\ lE_{elec} + l\epsilon_{mp}d^4, & d > d_0 \end{cases} \end{aligned} \tag{4.7}$$

E_{Rx} is an amount of energy to receive this message:

Table 4.3: Simulation environment I

Type	**Parameter**	**Value**
Network	Network grid	From (0,0) to (100,100)
	Initial energy	2 J/battery
	Head length	25 $byte$
Radio model	E_{elec}	50 nJ/bit
	ϵ_{fs}	10 $pJ/bit/m^2$
	ϵ_{mp}	0.0013 $pJ/bit/m^4$
	Threshold distance (d_0)	$\sqrt{\frac{\epsilon_{fs}}{\epsilon_{mp}}} \approx 87.7m$
GCC	Pulse signal length	56 $bits$
	Oscillating function	$f(\phi) = \frac{1}{b}ln[(e^b - 1)\phi + 1], b = 3$

$$E_{Rx}(l) = E_{Rx-elec}(l) = lE_{elec} \tag{4.8}$$

In the model, the radio dissipates E_{elec} to run the transmitter or receiver circuitry and E_{amp} for the transmitting amplifier. If the distance is less than a threshold d_0, the free space model (ϵ_{fs} and d^2 power loss) is used; otherwise the multipath (ϵ_{mp} and d^4 power loss) is used. The values of these variables are labeled in Table 4.3, and here $l = 56$ bits for GCC.

Here two ratio values are used to measure the node consumption situation and cluster number veracity. ECT denotes the energy consumption ratio which is the rate that average energy cost for each node to its maximum energy ($2J$). ERT denotes the cluster number error rate that is defined as $ERT = |m' - m|/m$. m is the expected final cluster number according to Equation (3.5) and m' is the real cluster number obtained from the simulation. Figure 4.11 shows the ECT and ERT of 300 nodes with different delay values τ under different coupling strength ϵ. All the resulting data appearing in this figure and Figure 4.12 are averaged over 500 network topologies and initial

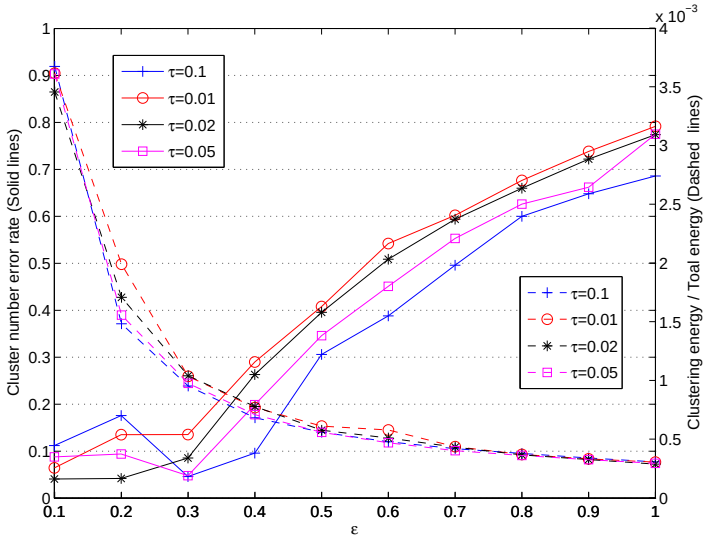


Figure 4.11: GCC energy consumption and cluster number error rate with respect to coupling strength ϵ.

phase realizations.

In Figure 4.11, dashed lines show that the energy cost ratio decreases with the increase of the coupling strength. When $\epsilon > 0.3$, the energy consumption ratio keeps under 0.1% (right axis), and the difference in cluster numbers have little effect on the energy cost. Solid lines show that the cluster number error rate increases with the increase of ϵ. Roughly, when $\epsilon < 0.4$, the error rate can be kept under 30% (left axis). When the cluster number is big ($\tau = 0.01, 0.02$), choosing the coupling strength to be as small as possible is preferred for obtaining expected m. While the cluster number is not big ($\tau = 0.05, 0.1$), choosing some proper strength ($\epsilon = 0.3$) can obtain the most exact m. According to the empirical data and results in Figure 4.11, the energy cost ratio for clustering below 0.1% is proper. Therefore, ϵ choosing inside the range $[0.3, 0.4]$ is suitable for both energy cost and cluster number veracity in this environment.

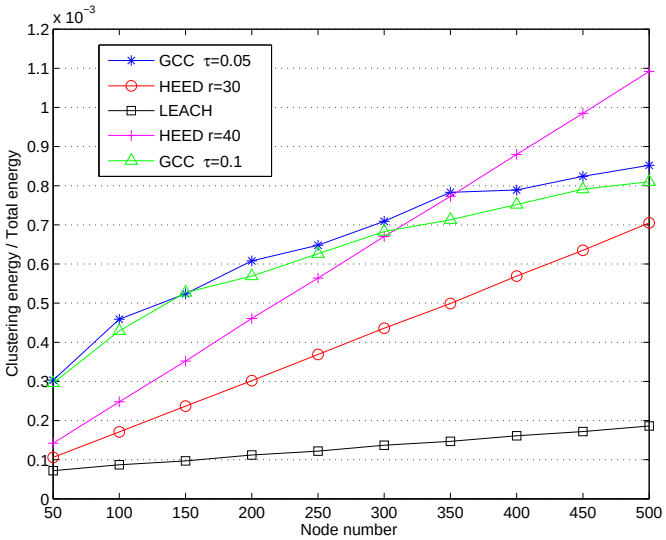


Figure 4.12: Energy cost ratio comparison under different node density. (GCC $\epsilon = 0.4$; LEACH 5% nodes as head)

4.5.4 Energy comparison with other clustering method

Next the energy consumption situation of the GCC method and the traditional clustering algorithms are compared. For HEED and LEACH, the broadcast packet used in the clustering procedures usually takes the size of 25 bytes = 200 bits. Figure 4.12 shows the comparison of ECT of the three clustering algorithms from sparse to dense network environment. For HEED, the fixed power level is used for intra-cluster communication and choosing min-degree as the second clustering parameter [87]. In GCC, the coupling strength uses the empirical value $\epsilon = 0.4$.

For GCC, energy cost increases with the node density and the different delay values almost have no influence. For HEED, however, energy cost linearly increases with the node density, and bigger transmission radius r costs more energy. This is because HEED chooses the best suitable node (highest residual energy, minimum cost) as head node from its neighbors. When the density increases or neighborhood enlarges, it has to communicate and check more neighboring nodes to decide the head node, and this process consumes more energy. Comparing HEED with

GCC, when transmission radius $r = 30$ and $\tau = 0.05$ HEED and GCC both produce around 10 clusters. Every node in GCC costs more energy than in HEED. At $r = 40$ and $\tau = 0.1$, they both produce around 5 clusters. In sparse network, GCC still costs more than HEED, but as the node density increases, GCC's ECT gets closer to HEED and becomes lower than HEED when $n > 300$. This effect is because at the latter stage of converging, the times of receiving action decrease due to more and more converged nodes. GCC's computing complexity is slower than linearly increasing. Therefore, the trend exists as when the network becomes more dense, GCC's energy performance becomes better. Here we use the most simple version of LEACH, in which the cluster head nodes are chosen based only on the rotation probability. Neither the residual energy nor other requirements are taken into consideration. It consumes the least energy among the three algorithms and increases mildly with the increase of the node density. In reality, it will cost more than that amount, but from the energy perspective, LEACH is still superior to HEED and GCC because of its simplicity.

Figure 4.13 and Figure 4.14 draw the total construction costs of HEED and LEACH with the global time synchronization protocol FTSP respectively, that they both get the equal multi-function like GCC does; the figures demonstrate that as a multi-functional method, the construction cost of GCC is quite efficient.

4.5.5 Stability for period elongation

The GCC method has an acceptable energy cost for clustering, and it can be improved further by the period adjustment stated in Section 4.3 Step 4.

Figure 4.15 shows the system converging procedures and the stability of clustering results when the oscillating period is enlarged. The Y axis records the numbers of superposed firing signals and the X axis represents their firing time points. The higher a line means the more signals are emitted at this time point. The same gray degree (color) in the figure represent the same iteration round. In the figure, nodes fire almost all the time in one round at first, gradually

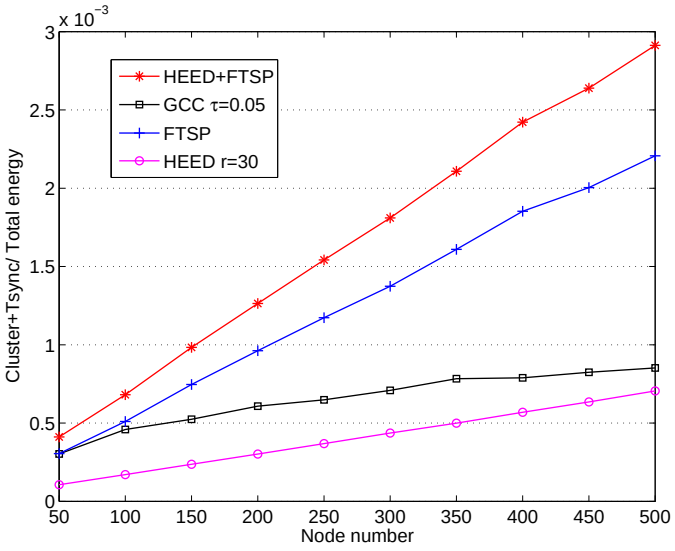


Figure 4.13: Total construction cost comparison between HEED+FTSP and GCC with the similar clustering and time synchronization results.

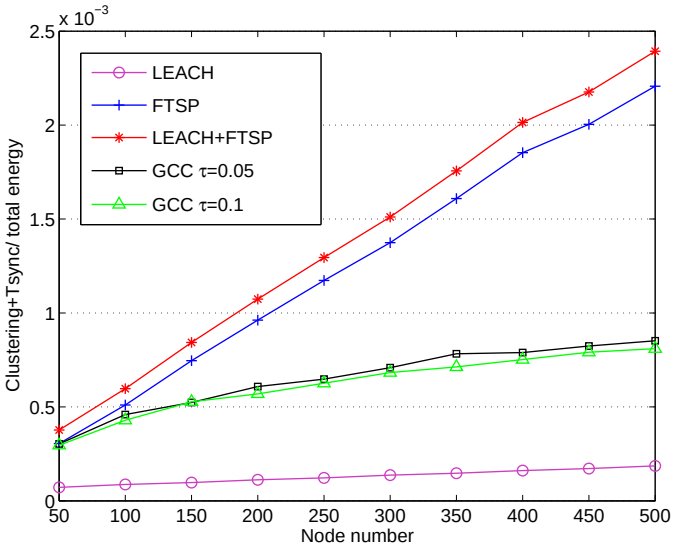


Figure 4.14: Total construction cost comparison between LEACH+FTSP and GCC with the similar clustering and time synchronization results.

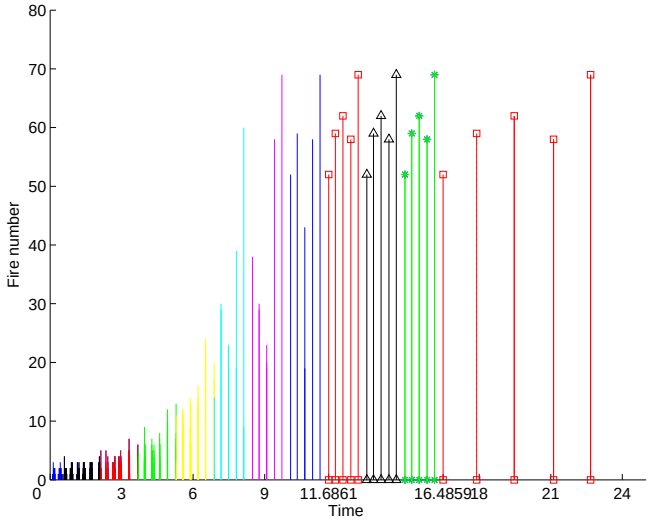


Figure 4.15: Fire strength along converging time and extended period after stable. ($\epsilon = 0.35, \tau = 0.1$)

there becomes fewer fire spots, while the fire numbers at these spots become bigger. Up until the 9th round (t=11.6861), the whole system converges into 5 clusters that fire alternately with approximately the same strength and intervals in one period. Once the system is converged, it will keep the stable clustering status periodically. After three periods (t=16.4859), we extend the system period to 5 times longer ($5T$) by slowing the oscillator phase increasing speed by 5 times (the original 5 ticks can be regarded as one). Each cluster keeps the same members and they are still phase-synchronized while their fire intervals are 5 times enlarged. This result testifies to the stability of GCC and shows that the period elongation is a feasible way to accommodate the different requirements between constructing and utilizing stages.

4.6 Summary

This chapter proposed a General Criteria-based Clustering (GCC) method for wireless sensor networks. By treating the sensor nodes as homogeneous oscillators and imitating the coupling

activity among neuron oscillators, GCC can cluster the nodes in both physical and logical ways according to any selected clustering criteria.

GCC overcomes the limitations in the existing clustering techniques by considering the nature and characteristics of sensor nodes. In particular, application level requirements can be used to influence the results of clustering. The use of the application level requirements leads to a connection between applications and clustering. On demand and dynamic changes in the applications may also be reflected in re-clustering. We believe that this is the first method to perform clustering based on general application information. Additionally, our method allows capturing the logical relationships that exist in distributed sensor nodes and representing those relationships in the form of Logical Clusters (LCs). Aside from clustering, the GCC method also produces an advantageous result in that the nodes within a cluster are phase-synchronized and the nodes among different clusters are phase-desynchronized. If needed, these synchronized/desynchronized phases can also be reversely mapped as the sensor nodes' logical clock, which enables applications that need time synchronization or desynchronization without requiring any additional algorithms or procedures. The next chapter will show how these sync/desync results help to realize the high efficient data communication in logical clustered networks.

Chapter 5

Data Communication Framework on Logical Clusters

Since GCC uses synchronization and desynchronization results that define the logical cluster, there is no cluster head elected or decided before or after forming the cluster. Due to the total self-organization of the clustering procedure, a node in the cluster is unaware of any other member nodes. Aside from non-identification and no head issues, the LCs partitioned in the phase domain may overlap in physical domain, which could cause serious interference problems under wireless transmission conditions. Moreover, the active area of logical cluster has more diverse shapes than the standard circle or polygon clusters, which make the head centered star topology no longer suitable for the cluster's interior communication. How to identify each formed logical cluster in the physical network area and manage the data inter- and intra- communication of these clusters are the subsequent necessary issues.

To actualize reliable data communication on logical clustered networks, the nodes that belong to the same logical cluster should firstly identify each other so as to construct connections among themselves; secondly, the overlapped logical clusters should be separated in a certain fashion, and the communication structure inside clusters should be able to adapt to any possible shape of LCs.

5.1 Logical Clustered Network

The General-Criteria Clustering is different from the normal clustering method from the basic signal form to the final clustering results. Before discuss data communication, the network condition after the clustering should be further clarified.

(1) Inside the logical cluster, the member nodes oscillate synchronically and keep the same phase value. However, each node does not know other members' locations or any other information, and there is no head node inside the cluster.

(2) Different logical clusters are phase-desynchronized. Between each pair of clusters, there exists a fixed phase offset. In each system period, clusters fire alternately.

(3) Each cluster fires once in one period and the emitted global pulse signals are used to maintain the clustered network status. The system period could be adjusted by the period elongation scheme (described in Section 4.3, Step 4) to be long enough not to interfere with local data communication.

(4) The signal used in the data communication stage is different from the one in the clustering stage. When doing the clustering, the node's firing signal is pulse-like and can be received by all nodes in the network (all-to-all coupling). These firing signals do not carry any node data but just indicate the firing time points. However, for data communication, the transmission message needs to carry data information with certain length. For convenience, in this paper the normal Radio Frequency (RF) message is used in the wireless sensor network to perform the data communication.

(5) Because the nodes are partitioned based on their properties and not locations, the shape of a logical cluster is diverse. Also, the physical fields of different logical clusters may overlap with each other.

5.2 Data Communication Framework on Logical Clusters

This chapter proposes a data communication framework for the logical clustered wireless sensor networks that focuses on the discussed issues above. The framework consists of the following two components. The first component builds a desynchronization supported quasi-TDMA (Time Division Multiple Accession) working/sleeping schedule among logical clusters. By making the different logical clusters alternately turn to active as scheduled, the overlapped logical clusters are spread in time scale. This set up not only eliminates the interference from the other clusters but also allows the member nodes that belong to the same logical cluster to identify each other. The second component constructs a novel Backbone tree in each logical cluster. The Backbone tree connects the scattered member nodes in the diversely shaped logical cluster and builds a routing plan for each node, which allows the data from every individual node to be transferred with well-balanced load and energy consumption.

Data communication in the sensor network is mainly used for bottom-to-up environment information collecting to the sink or Base Station (BS) in which further processing and analysis can be done and seldom up-to-bottom data query or control message dissemination. In this paper, we use “data gathering” - a generic sensor network application - as the illustration to discuss the data communication and associated simulation test in logical clustered networks. This method is used not only due to its commonality but also because it contains almost all types of data communication cases in the sensor networks: periodical data collecting from multi-nodes to the single sink node, message routing from a certain source node to the destination node, and command broadcast or convey from one node to multiple nodes.

In the clustered networks, the communication is divided into two layers: the communication between individual nodes inside each cluster and the communication between the head nodes of clusters and the sink. Corresponding to the two layers, two working components are proposed to construct a data communication framework for the logical clustered sensor network (Figure 5.1).

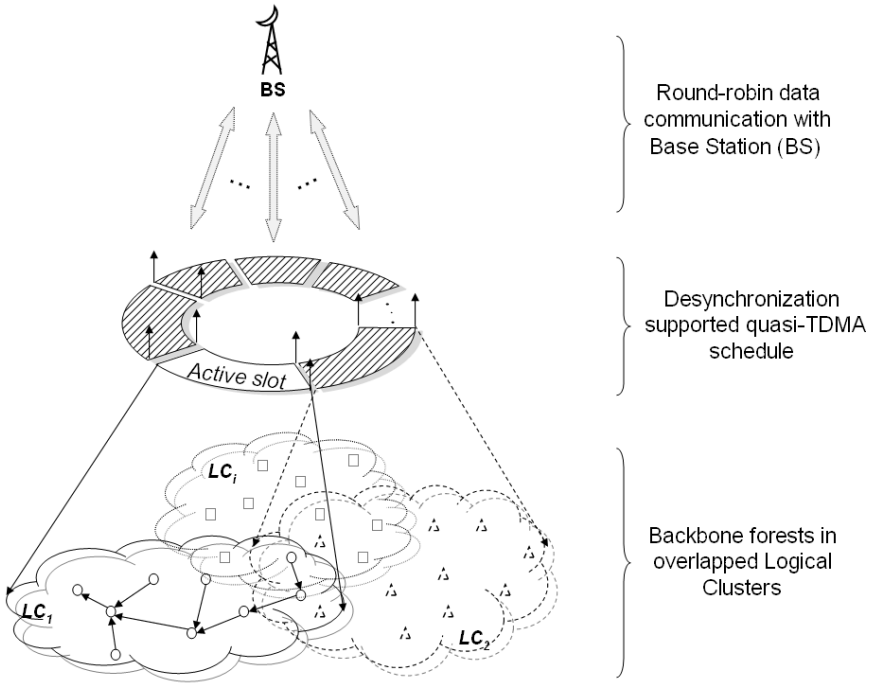
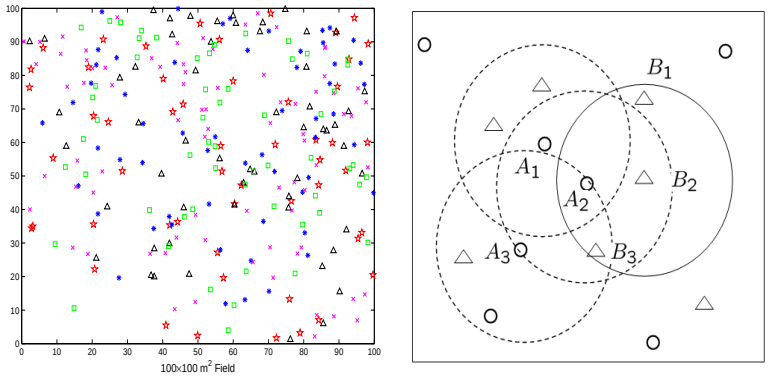


Figure 5.1: The data communication framework for the logical clustered wireless sensor network.



(a) Field overlapping. (b) Communication interference.

Figure 5.2: Overlapped Logical Clusters

5.2.1 Desynchronicity supported quasi-TDMA schedule between LCs

As discussed before, if the overlapped logical clusters do the transmission at the same time, the exclusive operations or communications among the cluster members will be seriously interfered with due to the physical communication radius (Figure 5.2). To eliminate the interference from different clusters and to identify the logical cluster members, a quasi-TDMA working/sleeping schedule is constructed between the logical clusters, which is naturally supported by the network nodes' synchronicity and desynchronicity characteristics inherited from the preceding logical clustering work. The nodes are scheduled so that even though they could physically communicate with each other, the topology is logically pruned to avoid it [63].

After the clustering, the nodes in the same logical cluster oscillate synchronously such that they fire and emit the pulse signals together when reaching the phase threshold. The group firing signals are global and strong enough to be detected by all the nodes. On the other hand, the different logical clusters are phase-desynchronized in that they fire alternately in period and emit global pulse signals sequentially (Figure 5.3). The signal sequence divides each oscillating period T into cluster numbers of time intervals. The interval between two successive pulse signals indicates the phase difference between the two clusters and its length. Although the phase offsets between each

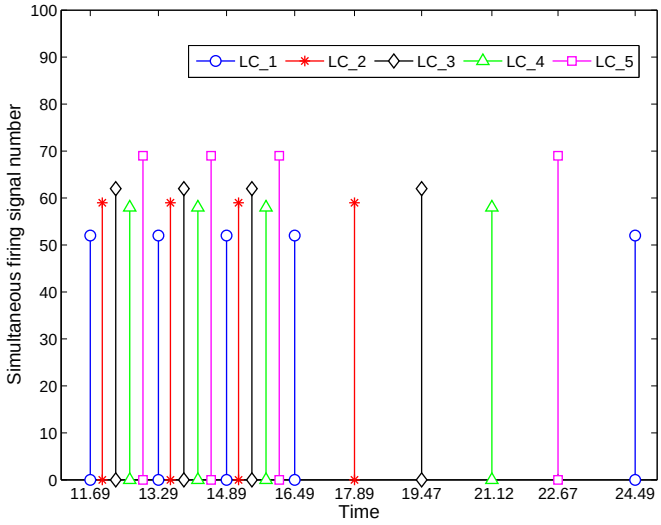


Figure 5.3: Regular pulse sequence emitted by logical clusters and five times period elongation.

pair of clusters are fixed, they are not the same size.

These self-divided cluster numbers of similar, length-fixed time intervals in one period realize a quasi-TDMA working schedule among the logical clusters. Each time interval could be assigned to one logical cluster as its private working time slot in the period and the sequential global firing pulse signals work as the natural slot division marks. By checking the bordering firing signals of their slots, the logical cluster members could automatically wake up and shut down collectively without the control of any coordinators. However, in the system, all the firing signals are the same. To simplify the control and to reduce management costs, the cluster usually chooses the interval neighboring its own firing point. Figure 5.4 shows three time interval belonging directions.

(1) The first case is also the simplest one in that the cluster chooses the interval after its own firing point as the active slot. The cluster members turn to the active state after emitting the firing pulses, then fulfilling the data transport duty, and when receiving the next system firing signals they will turn to the sleeping state. During sleep, the nodes could totally shut down their receivers and only keep the oscillating rhythm. Until it fires again, the cluster wakes up and enters the

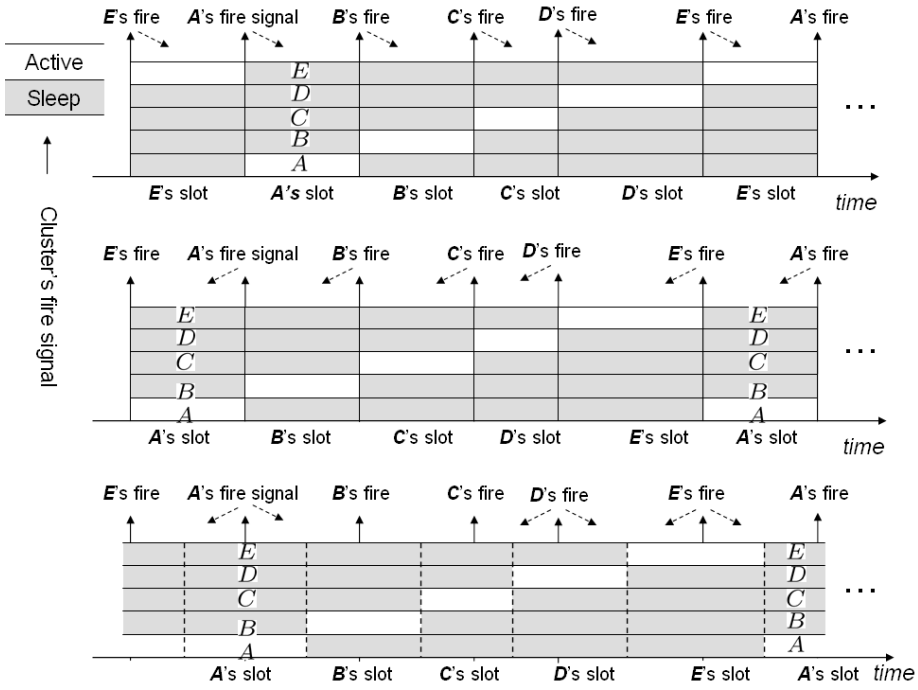


Figure 5.4: Working/Sleeping schedule and three time interval belonging directions.

active state again. In this case, nodes do not need to listen to the channel or keep any memory to recognize their own working slots' borderings.

(2) The second case chooses the interval before the cluster's own firing. The node wakes up as it receives the last preceding firing pulse signal before its own firing, which requires the node either to listen to and count the number of system firing signals before itself in one period or to memorize its phase value when the last preceding group fires. The node turns to sleep after it emits the firing signals that work as the ending mark of its working slot.

(3) The third case is usually designed for adjusting the active time evenness of clusters. The logical cluster chooses the part between midpoints of two successive intervals as its working time slot. In this case, the cluster members have to memorize their starting and ending phase values in the oscillating period so as to automatically start and finish the active state. The system pulses in this case are no longer the slot division marks and are only for maintaining the stable clustering state of the network.

Observed from the simulation, the pulse strength of a group is roughly proportional to its following phase interval length. This result is also explainable from the theoretical level that the more oscillators fired simultaneously, the larger push strength they can cause to enlarge the phase distance between the just fired group and the next firing one. The pulse strength of a group corresponds to the node number of this cluster, which also corresponds to the working load or data amount in the cluster for most cases. From this point onward, the first slot assigning direction is the most favored, in which the group that contains more members automatically gets a longer active time slot. Similar to the above discussion, the management of this case is also the simplest with the lowest cost. Besides the working load and active time balance, the working schedule takes several measures to make sure that each cluster has enough slot time to fulfill its sensing and transmission work. Firstly, clusters could extend their active time slot by prolonging the whole network oscillating period. Within the elongated whole period, each firing interval is also proportionally prolonged as showen in Figure 5.3. Secondly, the data fusion or aggregation mechanisms

are also used to reduce the number of messages and transmission time in the network.

Following the schedule, logical clusters take turns entering the active state. In network field, only the same logical cluster members keep awake such that the nodes can identify their colleagues. Meanwhile, all the other clusters are in the sleeping state where, except for the periodic oscillator firing, where nodes keep silent and shut down their receivers without using the public transmission channel. This situation effectively restrains the interference from the overlapping logical clusters, meantime mitigating the energy consumption of sleeping nodes. When the active time slot ends, the cluster turns to the sleeping/silence state and allows the next cluster to be activated.

The sleeping/working schedule has three fold functions: (1) identifying the same LC members; (2) eliminating the communication interference between different LCs as well as reducing transmission collision; and (3) conserving energies.

Because the whole sync/desync characteristic is inherited from the preceding clustering work, it does not require more specific coordinator or additional synchronization operations that are necessary in the normal TDMA scheme. Aside from the interval belonging direction guidance, the system does not need to spend any energy on constructing or managing this working/sleeping schedule; it is totally self-organizing and light-weight.

5.2.2 Backbone Tree supported data routing inside LCs

The member nodes of the same logical cluster can now be recognized, and they are possible to locate anywhere in the network. As demonstrated in the numeric test of GCC, the physical shape of logical cluster is so diverse that it could be polygonal, concentric circular, parallel strip, or even amorphous. Furthermore, there is no preassigned cluster head and no node knows the complete location of any others in the logical cluster. To collect the cluster nodes data or communication within logical clusters, the normal single hop head centric star topology inside the cluster is no longer suitable.

For the network field with an irregular physical shapes, constructing nodes into a tree topology as the data routing structure is a proper choice, since a spanning tree can self-grow along the field shape and connect all member nodes in the cluster. However, the root node of a tree is prone to be overburden due to high working loads and distant transmission to the base station (Figure 5.5(a)). Once the root dies, the whole network will collapse. To avoid the root node dying early, some literatures propose to construct the tree on the root of some super node, like the base station or sink node [41,69]. However, if the BS locates far away from the network, incorporating it as a part of the tree is not feasible, and this set up may also cause the hot spot problem in the neighborhood of the root. Another option is to change the root node and to frequently reconstruct the tree. This option will inevitably increase constructing time and energy costs. Based on this problem, Lindesey and Raghavendra [43] proposed the "Power-Efficient Gathering in Sensor Information Systems" (PEGASIS), in which all nodes are constructed into a linear chain. Data flow along the chain to arrive at the head, which is acted on by one of the chain nodes, and the head location changes in each turn (Figure 5.5(c)). The system can rotate the head duty without reconstructing the whole topology, thus reducing the amount of energy spent per round. However, putting all network nodes into one linear chain would cause significant data transmission latency.

Combining the characteristics of the spanning tree with linear chain topologies, and following the work [53], a Backbone Tree structure for logical clusters is proposed. There are two kinds of nodes in the tree: bone nodes, which connect to each other and form a chain-like backbone; and child nodes, which connect themselves to the nearest bone nodes (Figure 5.5(b)). The backbone chain in the spanning tree is like a 'Bus' traversing the network field, and its affiliated branches cover all the nodes in the network. To guarantee the dominating role of the backbone and to shorten the transmission latency of the tree, each bone node has at most two layers of child nodes. When doing the data communication inside the cluster, the data from the source nodes are firstly gathered to their bone nodes, then transported along the backbone chain to the destinies that can be the head or other normal child nodes. In the Backbone tree, the head is acted on by one of the

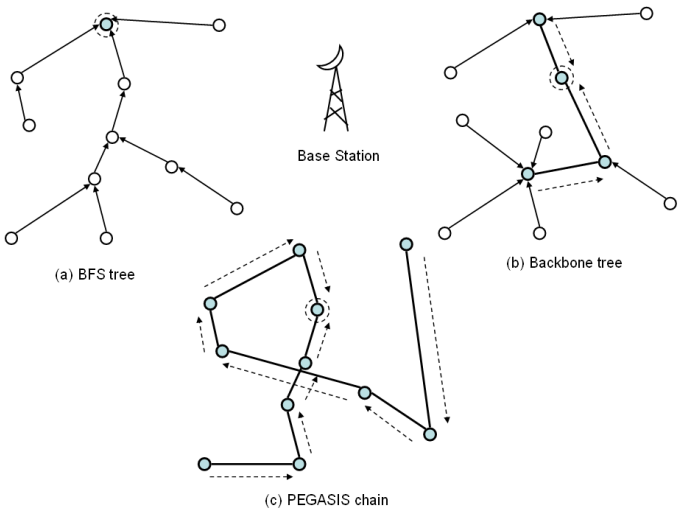


Figure 5.5: Demonstration of three topologies. (Dashed circled node means the current head duty location)

bone nodes and the duty shifts along the backbone chain each round (Figure 5.5(b)). Since the bone nodes take more responsibility than the child nodes, there is an energy threshold requirement for the bone node candidates when constructing the backbone; the value of this energy threshold is dynamically adjusted along with the network residual energy level. The detailed threshold value refreshing method is explained in a later section.

The advantage of constructing a backbone in a tree is that the network can rotate the head duty without reconstructing the whole tree. Moreover, the Backbone tree shortens the length of the chain by allowing the bone nodes to recruit child nodes, on which the nodes can transmit data parallel and ease the long latency problem. In addition, the Backbone tree can suit all diverse logical cluster shapes. The concrete construction algorithm of the Backbone tree will be presented later in Section 5.3.

5.2.3 Relationship of two components

There is a dependent relationship between the two components of the framework. The Backbone tree's construction in the logical cluster depends on the successful realization of the quasi-TDMA sleeping/working schedule among the clusters. Without the sleeping schedule, the node may not easily identify its logical cluster members, which complicates constructing the routing structure inside the logical cluster. Reducing the signal interference is preferred in any stage of construction and operation. Therefore, before constructing and each reconstructing of the Backbone tree, the first and second fold functions of the working/sleeping schedule are performed. Once the communication routing path inside the cluster has been built, each node keeps the identification numbers of its parent and child nodes in tables. Except for the packets from its child nodes, the node does not receive any other data packets; in the same way, the node only transmits data packet to its parent node. Based on this set up, the logical cluster member nodes maintain the memories of their colleagues. During this stage, the second and third-fold functions of the working/sleeping schedule conserve energy and contribute to eliminating the transmission interference among different clusters.

In the framework, the two components cooperate with each other to accomplish the communication of network data or other control information. On the other hand, although the two components are proposed based on the special characteristics of the logical cluster and work dependently in the framework, they can also be independently used for other practical purposes.

5.3 Construction of Backbone Tree

5.3.1 Systematic algorithm

Single tree

A backbone tree's construction has two stages. In the first stage, the bone nodes are self-determined and their first-level child nodes are subsequently recruited; the second stage starts after the back-

bone stops growing, in which the second level child nodes are connected to the tree. Before presenting the concrete constructing steps, some preconditions are stated here:

(1) Each node in the network has a unique identification number (ID).

(2) The sensor node can generate different transmission ranges by adjusting the transmission power level, but a basic transmission radius r is used for the child node searching and recruiting.

(3) All nodes keep a marking field to indicate its role in the tree or the connecting condition, and before the construction, all nodes are unconnected (s.mark=0).

(4) Every node knows the energy threshold ths of being a bone node, which is initialized by the first time hardware tuning or per broadcast message from the base station.

The Backbone tree construction step goes as follows:

Step 1 Randomly select one energy qualified node as the initial root that also works as the first bone node of the backbone; hence mark it as the root node (s.root=1).

Step 2 For each new bone node, recruit all neighboring nodes within the transmission range r as its children. Record the bone node as the parent of these child nodes and change their connecting state as s.mark=1.

If there is no neighboring node found for the bone node, go to Step 5.

Step 3 For each recruited child node in Step 2, and if its energy is qualified, find its unconnected neighbors and report the neighbor number to its parents node.

If there is no qualified child node, go to Step 5.

Step 4 If the parent of those qualified nodes in Step 3 is root, choose two child nodes that have the most neighbors as the new bone nodes (s.bone=1).

If their parent node is a normal bone node, choose one child node who has the most neighbors as the new bone node and set s.bone=1.

Send appointment messages to the new bone nodes and return to Step 2.

Step 5 The bone node, which cannot find qualified children in Step 2 or 3, sends End message to its upper bone node. Each bone node that receives the End message relays it to the next

node in the backbone chain and so on. Meanwhile, check the end flags (s.end) of these bone nodes.

If the bone nodes' end flags are unmarked (s.end==0), mark them as s.end=1 and return to Step 2.

If the end flags have already been marked (s.end==1), go to Step 6.

Step 6 All the bone nodes send order to their child nodes to start recruiting the second level child nodes. The child nodes, which receive the order and are energy qualified, search the unconnected nodes with transmission range r, and mark the new recruited nodes as s.mark=2, meaning that the child nodes at the second level.

Step 7 For nodes that are still unconnected after Step 6, broadcast the connecting request with an enlarged transmission radius r and decreased energy threshold requirement ths. For any node that receives this request, if its residual energy is higher than ths and s.mark$<$2, then send back the acknowledgement. The requesting node records the first acknowledging node as its parent and changes the connecting state as s.mark=2.

Repeat Step 7, until all nodes have been connected to the tree. Then terminate the algorithm.

Figure 5.6 demonstrates the flowchart of the Backbone tree's constructing procedure. There are three main loops in the construction procedure. The first loop ('Y' branch for 'Has energy qualified child?') starts from Step 2 to Step 4, which describes the procedures to determine a new bone nodes on the backbone chain. The backbone starts from the root node and grows in two directions. The second loop (Step 5, 'N' branch for 'Has energy qualified child?') is used to check whether both directions of the backbone has ended. It only runs once, because the first running changes the system ending flag that makes the loop entering condition and cannot be met again. The iteration times of the first loop are the number of bone nodes on the backbone chain. After jumping out of the second loop, the system goes into the second construction stage (Step 6, 7). For the single tree in an integrated network area, the loop on Step 7 forces all the left nodes to be

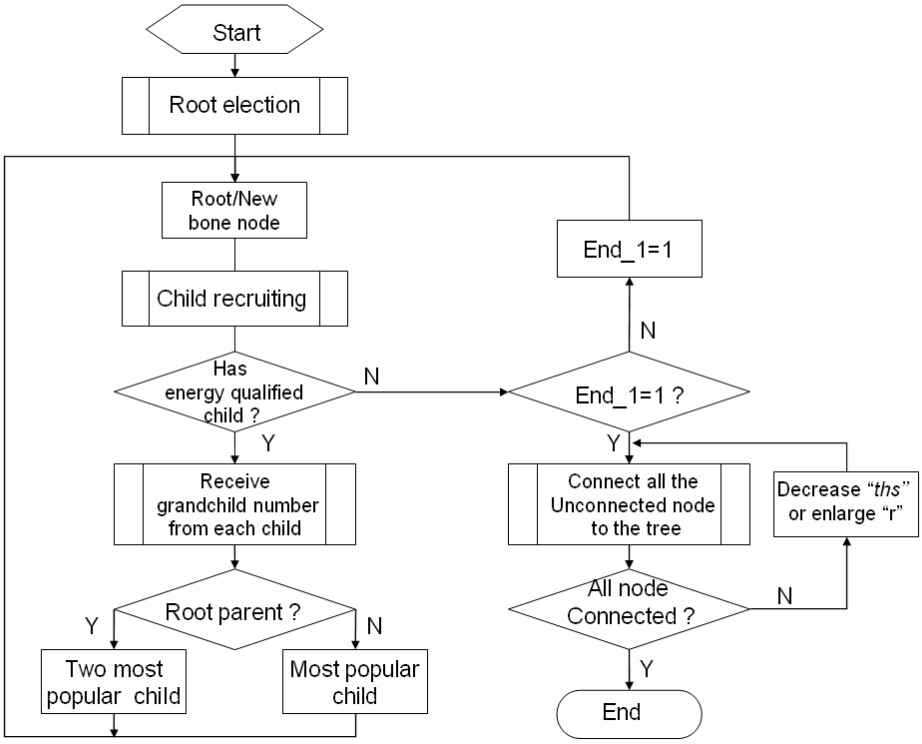


Figure 5.6: Flowchart of the Backbone tree constructing steps.

connected to the tree with gradually relaxed recruiting conditions. However, if the connection cost of these left nodes is too high, there arouses the multiple tree situation, which will be discussed in the next.

The above constructing steps and discussion show that the whole algorithm is totally self-organized, and it is also robust when some message lost causes losing the chance to compete the bone node. It is still not connected, so it can compete for the following bone node contending.

Root election

In the above construction Step 1, an energy-qualified root node needs to be initially elected. However, not all the active nodes in the network are energy qualified and neither the base station nor any single node knows the other nodes' energy situations. A straightforward way to decide a root is to let the energy qualified nodes nominate themselves, then all the members confirm one as the final root. The unique ID is used in the root claiming message. A CSMA_CA (Carrier Sense Multiple Access with Collision Avoidance) scheme is used to avoid the excessive claims at the same time. That scheme means that before the node broadcasts its claiming message, it should first listen to the public channel for a random time, and only if no root claiming message has been detected can it claims itself. This 'listen-before-speaking' scheme greatly reduces the claiming message number in the channel, but there still probably appears more than one root claimer because of the wireless transmission delay. A simple option is to default the one with the smallest (or biggest) ID as the final root. A more reasonable solution is to choose the one with the highest residual energy level, but this option requires the claiming messages to contain the nominators' energy information, which will consume more energy.

Multiple trees

For a logical cluster, there may exist more than one separate areas, which begets the possibility of constructing multiple backbone trees in one cluster. However, due to distributed characteristic and

the absence of global cognition of the network, whether or not to construct multiple trees and how to divide them become problematic.

If there exist more than one tree in one logical cluster, each tree will communicate with the base station independently. Therefore, in each round there is more than one distant transmission, which will increase the network's total energy cost. However, forcing the sparsely scattered nodes to join one tree with relatively larger radii may also cause higher energy consumption in the local transmission. The comparison of these two costs depends on the distribution of the node locations and the structure of the existing tree. Technically, the determination of the new tree's existence only needs to be done once, since the trees work independently after being constructed and their reconstruction periods are different. The node division of different trees will not change unless they are reconstructed at the same round. At the other time, these individual trees share the same active time slot of the logical cluster. Since the number of trees in one cluster is few, the multiple tree heads' communication with base station will not cause much collision.

Here a system parameter R is set as the critical radius for constructing or not the new tree. When the second layer children are recruited in Step 6, R is used as the biggest enlarged search radius. If some untouched nodes are still left in the logical cluster, the system determines they are located in separate areas and should construct one or more new trees. For the possibility of multiple trees in the logical clusters, the algorithm's Step 6 and 7 should be revised as:

Step 6 ... The child nodes, which receive the order and are energy qualified, search the unconnected nodes with transmission range R and mark the new recruited nodes as s.mark=2 meaning the second-level child node.

Step 7 Terminate the algorithm if all the nodes inside the cluster have been connected to a certain backbone tree. Otherwise, return to Step 1 and start a new tree.

Algorithm 2 and Algorithm 3 present the complete Backbone tree construction pseudo-code.

Algorithm 2 Greedy_backbone(int n, double ths)

1. $newtree \leftarrow 1$
2. **while** (connected node number $< n$) **do**
3. **if** ($newtree$) **then**
4. $p \leftarrow$ Initial-newtree(ths)
5. **end if**
6. **repeat**
7. **for** each ($v_i \in S\_nbr\_p$) AND ($E_{v_i} > ths$) AND ($s[v_i].mark = 0$) **do**
8. Find $S\_nbr\_v_i$
9. v_{max}: $|S\_nbr\_v_{max}| \geq |S\_nbr\_v_i|$
10. **end for**
11. **if** ($|S\_nbr\_v_{max}| \neq 0$) **then**
12. $p \leftarrow v_{max}$
13. $Bonenode\_declare\_msg(Node(v_{max})\_ID)$
14. Mark all the nodes in $S\_nbr\_v_{max}$ as $Connected$
15. **else if** ($end\_1 \neq 1$) **then**
16. $end\_1 \leftarrow 1$
17. $p \leftarrow root$
18. **else**
19. $end\_2 \leftarrow 1$
20. **end if**
21. **until** ($end\_1 = end\_2 = 1$)
22. **for** ($j = 0$ to $j < n$) **do**
23. **if** ($s[j].mark = 0$) **then**
24. Try to connect itself to the nearest qualified $Connected$ node in its range R
25. **if** ($Connected$) **then**
26. $s[j].mark \leftarrow 2$
27. **else**
28. $newtree \leftarrow 1$
29. **end if**
30. **end if**
31. **end for**
32. **end while**

Algorithm 3 Initial-newtree(double ths)

1. Randomly elect a ($E_i > ths$) AND ($s[i].mark = 0$) node as $root$
2. $Root\_declare\_msg(Node(i)\_ID)$
3. $S\_nbr\_root \leftarrow \{v : v$ is unconnected and within the range of $root\}$
4. Mark all the nodes in S_nbr_root as $Connected$
5. $newtree \leftarrow 0$
6. **return** $root$

Reconstruction and the threshold refresh

As the routing topology of the nodes, the Backbone tree's energy consumption on each node is not in equilibrium. To balance network energy consumption and extend its lifetime, the Backbone tree topology needs to be reconstructed in some frequency. Furthermore, along with dissipating network energy the energy threshold value ths for selecting the new generation of bone node also should be dynamically updated to suit the current network situation.

The Backbone tree's reconstruction period and the associated energy threshold refreshing method are related to the concrete data communication applications. Here we still use the periodical data gathering application as the example to discuss the Backbone tree's reconstruction. In each round, when bone nodes transport packets to the head, they piggyback their residual energy information. As a part of the final head packet, the energy information of all bone nodes is reported to the base station. Since the bone nodes usually consume more energy than the normal ones, a base station can estimate the lowest current energy level possible in the network. By setting the new energy threshold value higher than that in the next construction, the system can avoid always choosing the same group of nodes as the bone nodes. This method effectively balances the network node energy consumption, which is also verified in our later simulation results.

Therefore, before each reconstruction, the base station calculates the new bone node energy threshold value ths and broadcasts it to the whole network. The node that receives the new threshold value changes its connecting status as unmarked (s.mark=0) and enters the reconstructing

stage. The Backbone tree's reconstruction could be launched after all the current bone nodes have taken the head duty once. In this case the reconstruction period for each cluster is different, which is dependent on the length of backbone in each tree. The reconstruction also could be activated by the base station, making the reconstruction period uncertain.

5.3.2 Implementation in individual node

The algorithms in Section 5.3.1 describe the constructing procedure of Backbone trees from the systematic aspect. In this subsection, the constructing actions from the individual node aspect are presented. The data structure in memory, communicating mechanism, termination judgment and message format are given to implement the functions of Algorithm 2 and 3. This implementation focuses on a single tree's construction in general network topologies; its time and message communication complexity are also analyzed.

Action on each node

For not losing track, some steps are similar with the ones in Section 5.3.1.

Some preconditions are stated here:

1) Each node in the network has a unique identification number ID.

2) Each node can adjust its transmission range power level.

3) Each node keeps a neighbor table T_{nbr} to store the IDs of all its neighboring nodes located within the transmission range r. The content of this table is obtained from the BS message or constructed according to the nodes' mutually informing messages. Once the T_{nbr} of each node is constructed - unless some nodes die - the content will not change.

4) Each node also keeps a dynamic child table T_{d_cld}. Each time, before constructing the new Backbone tree, the node loads the content of neighbor table T_{nbr} into the child table T_{d_cld}. During the tree constructing procedure, the content of table T_{d_cld} is updated from time to time, until the node is connected to the tree. Figure 5.7 shows the structures of two tables and their loading

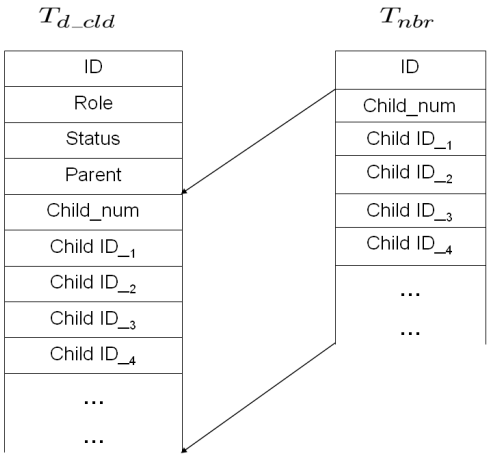


Figure 5.7: Structure of neighbor table T_{nbr} and child table T_{d_cld}.

relationship.

5) Each node records the energy threshold of being a bone node, which is refreshed termly by the BS broadcasting.

The following are the construction steps of the Backbone tree:

Initialization: All nodes load new child table T_{d_cld} from their neighboring table T_{nbr}, and reset their 'Status' field as $Unconnected$, 'Role' as 1, which means normal child node.

Step 1 Randomly select one energy qualified and unconnected node as the initial root and change its 'Role' field as -1, which means the root node. The root also works as the first bone node of the backbone B_0.

Step 2 For each new determined bone node, check if there are entries in its child table T_{d_cld}. If there are, then go to step 3; if are not, wait until the time out then go to step 4.

Step 3 For each bone node B_i that has a child, determine the next bone node B_{i+1}. If the current bone node is a root node, then determine two bone nodes from B_0. Return to step 2.

Step 4 For all unconnected nodes, broadcast connecting request to all nodes in its neighboring table T_{nbr}. If the first acknowledgement message comes from node j, change its 'Status' field as

$Connected$, 'Parent' filed as node j's ID and 'Role' field as 2, which means it is the second-level child node. For any node that receives this connecting request, if its residual energy is higher than the current threshold value, send an acknowledgement message to the request node.

Step 5 If the unconnected node in step 4 did not receive any acknowledgement message, then enlarge its transmission radius while decreasing the threshold requirement and rebroadcast the request message. This time, only the node whose 'Role' field value is less than 2 reacts to this request. The receiving node that meets both new threshold and 'Role' requirement sends back the acknowledgement message. Repeat step 5, until all nodes' statuses become $Connected$. Then terminate the algorithm.

In order to show how the current bone node B_i decides the next bone node B_{i+1} in step 3, below are the detailed actions of the bone nodes and the related normal nodes respectively.

For the current bone node B_i

1) B_i receives the appointment message and changes its 'Role' field as 0, which means normal bone node, and 'Parent' field as the sending node ID;

2) Assuming B_i has m_i entries in its current T_{d_cld}. B_i sends the grandchildren number inquiry request to all these m_i children.

3) B_i waits for all the m_i children's reply messages within the valid waiting time (VWT) and picks up the one that has the most grandchildren number.

4) B_i sends the appointment message to the picked up child and affirms it as the next bone node B_{i+1}.

For each node in the bone node B_i's T_{d_cld} table:

1) After receiving the grandchildren number inquiry from B_i, mark its 'Parent' field as B_i.

2) Broadcast the updating message to all nodes in its T_{d_cld} table, which claims it has been recruited and asks all the receivers to delete its entry in their T_{d_cld} tables.

3) Meanwhile, wait for the updating messages from other child nodes and delete the corresponding entries in its T_{d_cld} until the valid waiting time (VWT) ends.

4) After the waiting time ends, clean up its T_{d_cld} and count the new child number. If the node's residual energy is higher than the energy threshold value, then send the reply message to its parent node.

5) Change the 'Status' field as $Connected$. Unless receiving the bone node appointment message or latter connecting request in step 4 and 5, the node no longer receives any updating or inquiry requests.

Time and message exchange complexity

In the above algorithm, step 4 and 5 are to connect the residual nodes to the tree. To connect one isolated node, it may request all the other nodes in network, which costs $O(n)$ time. However, the unconnected node number is constant, so the total processing time to connect all the residual nodes is at worst $O(n)$. In the algorithm, step 3 is the most time consuming part, in which each node will be inquired at most once. To reply to the inquiry, it has to communicate with all its neighbors. Thus, the processing time for arbitrating the bone nodes is at most $O(n)$ for each node, and $O(n^2)$ for the whole network.

Next, the energy cost and message numbers used in step 3 are analyzed. For each child of B_i, in order to make it connected, there cost one receiving from its parent node, one reply to its parent, one broadcast to all nodes in its T_{d_cld} table, and at most m_i times receiving from the updating messages. Among them and besides the above mentioned cost, the chosen new bone node also needs to receive the appointment message once; one new appointment message sending; one inquiry request broadcast to all nodes in T_{d_cld}; and m_i times reply message receiving from each child. For the nodes that are not the children of B_i but receive the updating messages from the children of B_i, they just delete the corresponding entries in their T_{d_cld} tables and do not reply with anything.

During this procedure, the system uses four types of control messages: 1) Bone node appointment message, 2) Grandchildren number inquiry message, 3) Table updating message, and

Figure 5.8: The format of system general control messages.

4) Child number reply message. Among these, 1) and 4) are uni-cast that have specific destination IDs; 2) and 3) can be whole transmission area broadcasts or multi-casts with multiple destination IDs. The Figure 5.8 shows a general network message formate.

The 'Type' designates the property of message which is either a control message or data message. Only if it is control message will the following 'Subtype' and 'Type related data' fields have meanings. The 'Subtype' specifies the kind of control message that can be one of the above four or the connecting request or acknowledgement message used in step 4 and 5. And the 'Type related data' field content is related to the 'Subtype' field value. Whatever the control message type is, the packet length is short and a small amount of energy is needed. If it is a data message with a long data length, more energy is necessary and the data information is attached in the latter part of message structure.

5.4 Global Working Mechanism of LC and Data Communication

Integrate the construction of Backbone tree within each LC with the working/sleeping schedule among LCs, Algorithm 4 demonstrates the complete working flow of the data communication on the logical clustered network.

5.5 Simulation Results

There are two main parts in this section. The first part demonstrates examples of the Backbone tree in the logical clusters that directly result from GCC. The second part testifies to the basic performance of Backbone tree with the comparison of its two inspired topologies: the linear chain

Algorithm 4 Main()

```
1. m ← Logical clustering( )
2. for (i = 1 to m) do
3.    Wake up the k members of the 'ith' cluster
4.    if (First_construction) OR (New_threshold(ths)) then
5.       for (j = 1 to k) do
6.          s[j].mark ← 0
7.       end for
8.       Greedy_backbone (k, ths)
9.    end if
10.   Application_execution()
11.   Turn to sleep
12. end for
```

of PEGASIS and the Breadth First Spanning (BFS) tree. It includes the transmission latency and the lifetime of the network under the application of data gathering.

5.5.1 Backbone trees in Logical Cluster

The logical cluster resulted from the GCC method may have various physical shapes. Therefore, we choose two kinds of representative cluster shapes to illustrate the adaptability of Backbone tree structure. The first is the open planar clusters (polygonal cluster or the ones formed by scatter nodes). The second is the physical restricted cluster (circular or strip shaped logical cluster). Figure 5.9, 5.10 and 5.11 show the successfully constructed backbone trees in the polygonal, concentric circular, and strip logical clusters respectively. To make all field nodes connected to one tree, the transmission radius is an important parameter, otherwise there may appear unnecessary multiple trees. With different transmission radii, the length and structure of the Backbone tree may has some difference. In Figure 5.9, 5.10 and 5.11, three hundred nodes are deployed in a

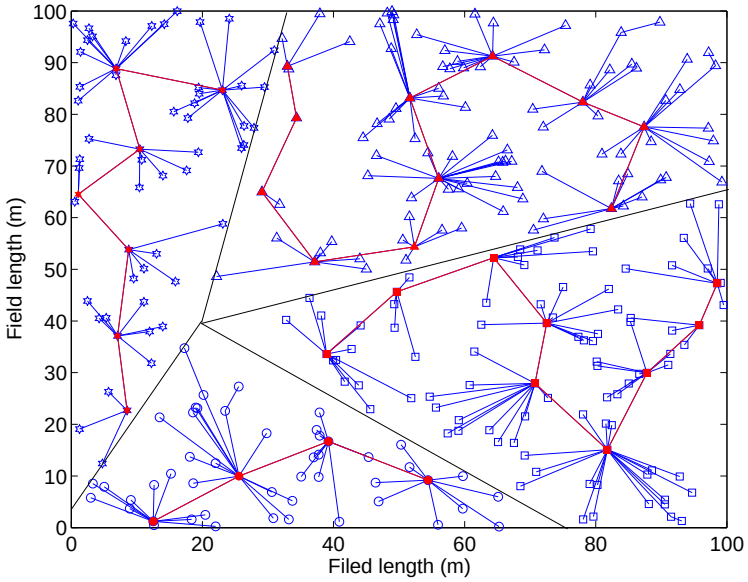


Figure 5.9: Backbone trees in polygonal logical clusters.

$100 \times 100\ m^2$ field, and the node transmission radius uses $r = 17$.

5.5.2 Backbone tree performance

Transmission latency

The transmission latency is measured by the worst case that may happen during the data transmission. For chain-like topology, all receiving and transmitting are serial on the chain. The best case is when the head duty flows to the midpoint of the chain; the latency is around half of the length of the chain. The worst case is when the head duty flows to one end of the chain, since the latency would be the whole length of the chain. Because most of the transmissions between cluster members are local, the major time consuming part is on each node's receiving, packing, and retransmission functions. Here we use the network relay times (hops) to measure the transmission latency.

In PEGASIS, all member nodes are constructed into one chain, and the length of the chain

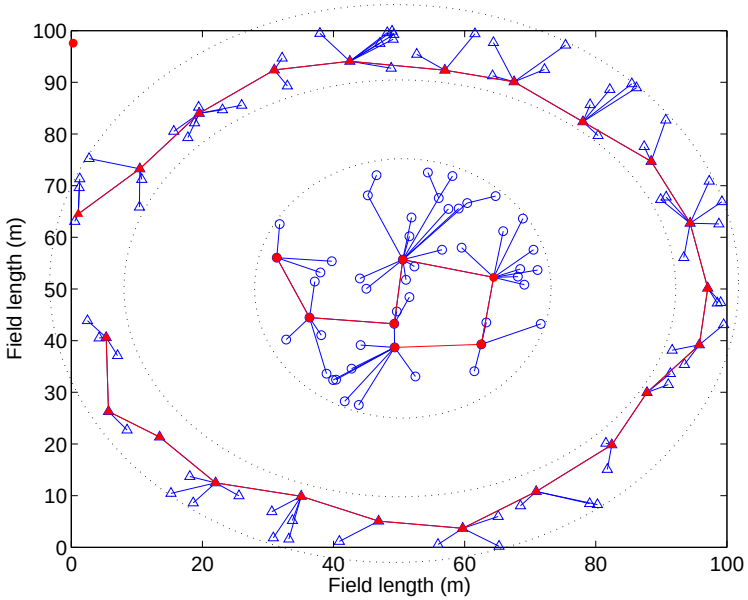


Figure 5.10: Backbone trees in concentric circular logical clusters.

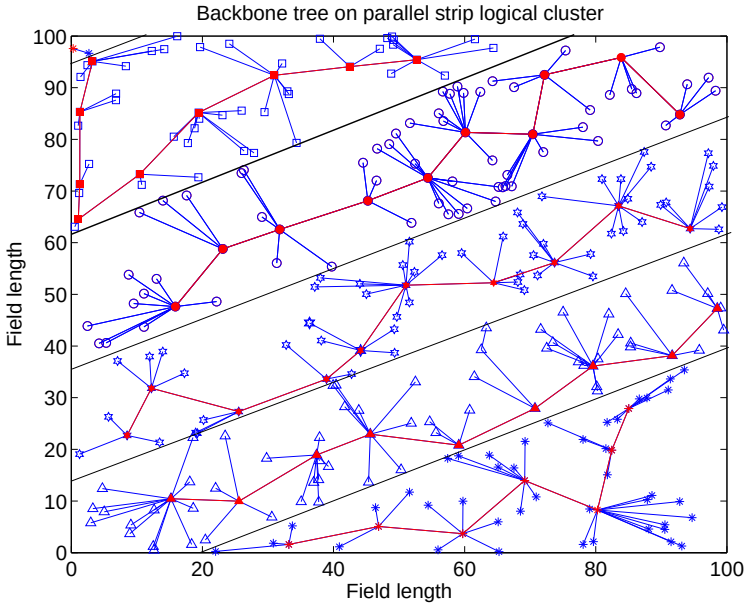


Figure 5.11: Backbone trees in strip logical clusters.

is the number of cluster member nodes. It has no relation with the transmission radius, so the transmission latency of PEGASIS is the number of cluster members and it does not change unless some node dies early. In the Backbone tree, the bone nodes have at most two layers of child nodes, so the longest latency of the backbone topology is the length of backbone chain plusing one. For BFS tree, the latency is the layer number of the deepest leaf nodes in the tree. For both the backbone and BFS trees, the final tree topology is influenced by the transmission radius. Here we do the transmission latency comparison between these three algorithms under different transmission radii.

Figure 5.12 shows the latency situation in the upright polygonal logical cluster in Figure 5.9 (n=95) and Figure 5.13 shows the latency situation in the larger circular logical cluster in Figure 5.10 (n=90). For the field size of $100 \times 100\ m^2$, the transmission radius ranges from 12 to $24m$, and all the data in the above two figures are averaged from 100 times tree constructions. The figures show that whether the cluster shape is open planar or physical restricted, the latency value of PEGASIS is always very high comparing with other two. The BFS has the shortest latency time due to its breadth-first construction rule. The latency of Backbone tree is a little longer than the BFS tree but much shorter than PEGASIS. In addition, for the different logical cluster shapes, the latency values in circular ones are higher than in polygonal shapes for both Backbone and BFS trees.

Network lifetime

In this part, we compare the lifetime of three topologies when operating periodical data gathering on them. For generality, the network is no longer set on some specific logical clusters but instead on a normal square field. To reduce the data volume, data fusion or aggregation is used. When a node receives messages from all its child nodes and the previous nodes on the chain, it fuses its own data with the received ones, repacks them as same-sized messages and then transmits. The final head node also transmits the same size message to the BS. We use the working round number

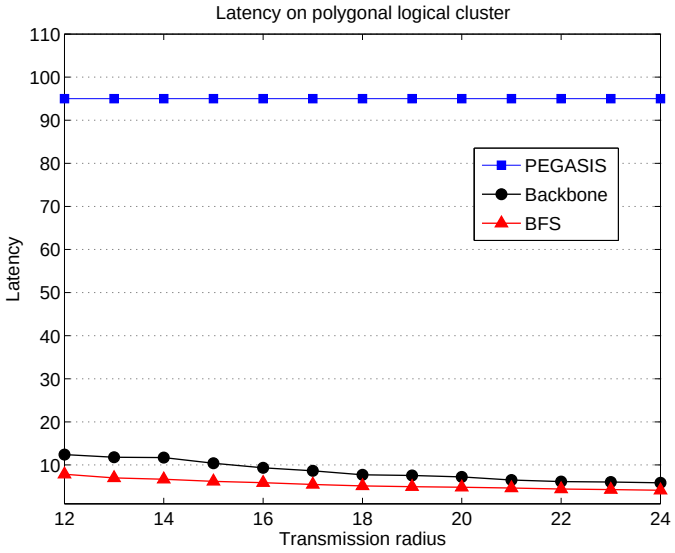


Figure 5.12: Transmission latency on polygonal logical cluster with different transmission radii.

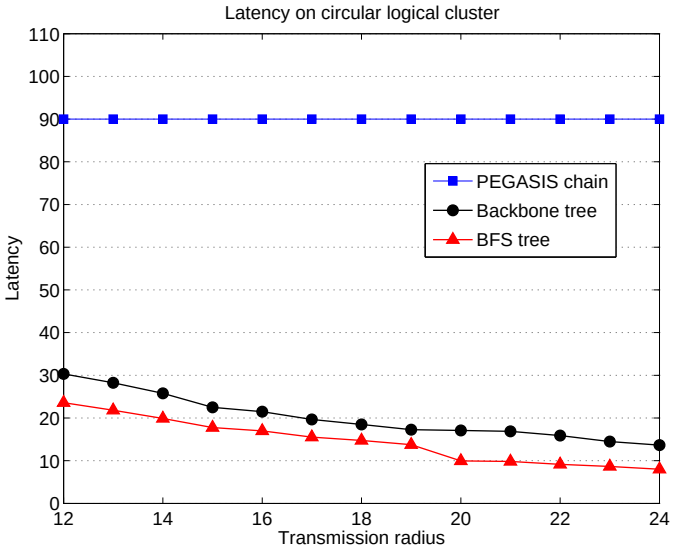


Figure 5.13: Transmission latency on a circular logical cluster with different transmission radii.

Table 5.1: Backbone tree network environment setting

Type	**Parameter**	**Value**
Network	Network grid	From (0,0) to (100,100) m
	Base station	(200,200)
	Initial energy	1 J/battery
	Head length	25 $byte$
	Message length	2000 $bits$

before the first and the early 10% nodes die to measure the network lifetime. Transmitting and receiving energy consumption follow the same first order radio model described by Equation (4.7) and Equation (4.8). Aside from the former variables's values listed in Table 4.3, new network parameters are listed in Table 5.1. All the data in the following four figures are averaged over 100 times network distribution.

Figure 5.14 shows the working round numbers of a network from the death of its first to the last node, which demonstrates the lifetime performance and the energy consuming trend of the three algorithms. In the figure, the later the first node dies, the better load balancing the algorithm has. Additionally, the later the last node dies, the better 'survival ability' the algorithm has. The Backbone tree shows good load balancing and distributed energy consumption. Looking at the death of the first 30% of the nodes, their lifetimes in the Backbone tree are much longer than in BFS tree. Also the first 16.7% of dead nodes' lifetimes in the Backbone tree are longer than or similar with PEGASIS. In the Backbone tree, after the first node dies, the whole network also dies rather quickly. This property is preferred by some applications that require the whole network to work cooperatively. On the other hand, although the first node dies early, the BFS tree shows a longer whole network lifetime than the other two. After a portion of nodes die, the remainder nodes can continue to work for a rather long time with relatively high energy levels. Choosing what kind of property depends on the application requirement. From the above two

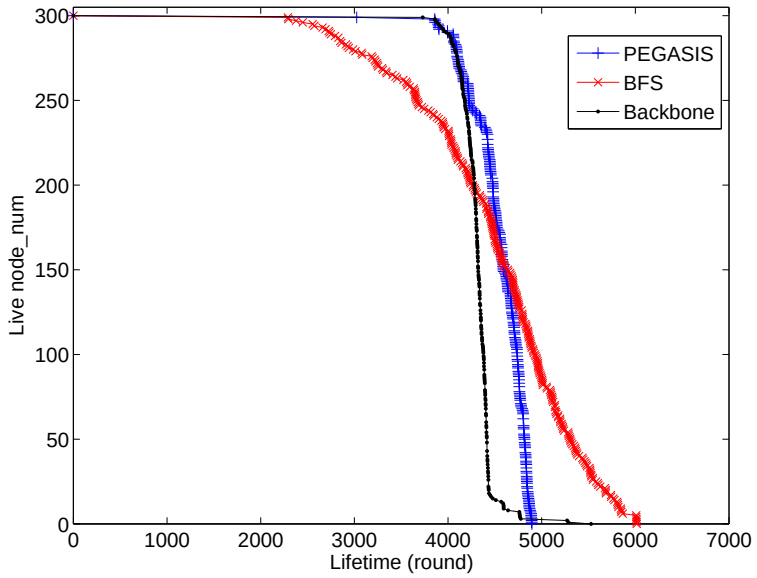


Figure 5.14: Lifetime of all nodes in the three algorithms. ($n = 300$, $r = 16$).

criteria, PEGASIS achieves the best performance among the three. Its first node dies later than the BFS tree, and it has a longer whole network lifetime than the Backbone tree. However, its long latency is fatal and restricts its extensive applicability.

Figure 5.15 shows the lifetime and the corresponding residual energy comparison of the three algorithms under different node densities when the first 10% of the nodes die - where the transmission radius is fixed as $r = 17$. The lifetimes of PEGASIS and Backbone tree are longer in dense networks than sparse networks. This circumstance occurs because the sparse network usually requires longer transmission distances between two nodes on the chain. The node density has no influence on BFS tree topology. Before the first node dies, the Backbone tree lives longer than the PEGASIS chain and BFS tree. This fact owes to the energy threshold when choosing the backbone node candidates. Here, we use the last round bone nodes' average residual energy as the new threshold. The root node selection in BFS can be used the same way, but the intermediate nodes cannot be selected by this method because of the construction rules of BFS tree. Therefore, although the BFS tree is reconstructed on different qualified roots every round, the intermediate

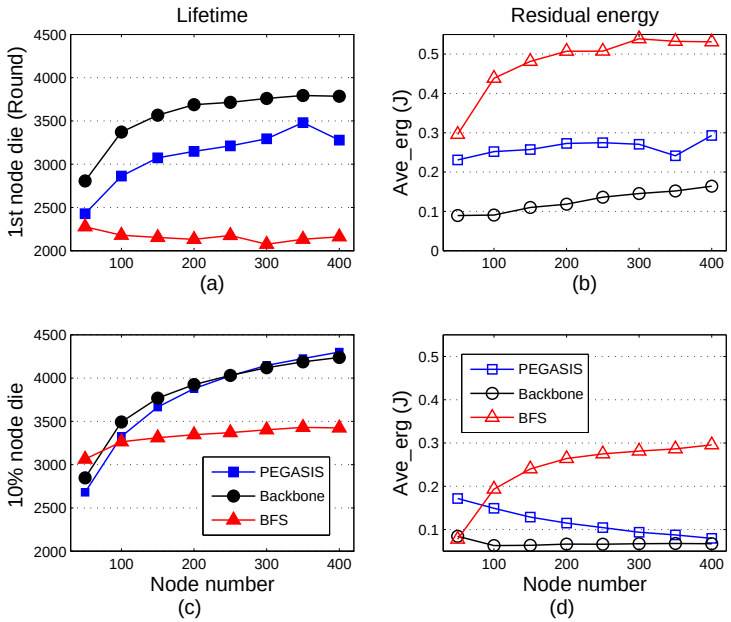


Figure 5.15: Lifetime and the corresponding residual energy comparison of the three algorithms under different node density when the first 10% of nodes die ($r = 17$).

nodes that have large numbers of child nodes may not vary. These nodes have a higher probability of dying earlier. For PEGASIS, each node only receives and transmits once in one round, but the distance between two continuous nodes on the chain may be long, especially at the later part of the chain, so the later part of chain may break first. Figure 5.15 (a) shows that the Backbone tree has good load balancing. When 10% of the nodes die, all three topologies can continue to operate. BFS gains the most increment on its lifetime, but it still lives shorter than the others. PEGASIS increases to the similar level as Backbone tree (Figure 5.15 (c)). Backbone tree obtains the least lifetime increment. From the asymptotic trend of the Backbone tree lines in Figure 5.15 (c) and (d), its lifetime will not increase much in the future because the averaged residual energy of all nodes is almost close to zero. Sending energy information and receiving BS broadcasting cost the backbone node some energy. The backbone node uses this energy to tradeoff for load balancing.

Figure 5.16 shows the lifetime and corresponding residual energy comparison of the three algorithms under different transmission radii when the first node dies and where the node number n equals 200. For the Backbone tree, when the transmission radius $r \leq 24$, the lifetime keeps almost a similar level. After $r > 24$, the lifetime decreases as the transmission radius increases. This situation occurs because in the Backbone tree - with the increases of the transmission radius - energy consumption of bone nodes also increases in two different aspects. Due to the increased r, the distance between two continuous bone nodes is increased as well, and the transport cost on the backbone becomes higher. The increase of r also makes the bone node number become smaller, and under the same network node number each bone node has to carry more child nodes than previously. This increase makes the bone nodes consume energy quicker, which is shown in the simulation results of the bone node average residual energy (star marked dash line in Figure 5.16 (b)). Since the Backbone tree is not reconstructed every round like the BFS tree, the bone node may die before the next reconstruction happens. The energy threshold control speed is slower than the bone node energy depleting speed. The different r almost has no influence on the lifetime performance of PEGASIS, because the algorithm is independent of the transmission radius. The minor fluctuations on the PEGASIS line is due to the different node deployments. For the BFS tree, the lifetime increases slightly with the increase of the transmission radius. The reconstructing of BFS tree on each round eases the heavy node burden problem caused by the longer r. Compared to the three algorithms' lifetimes before the first node dies, the Backbone tree is superior to both the BFS tree and PEGASIS chain topology under small to medium length of the transmission radii. When $r \leq 30$, the lifetime of Backbone tree becomes shorter than the PEGASIS chain but still longer than the BFS tree.

The situation of lifetime comparison when 10% of nodes die is shown in Figure 5.17. For the same reasons as explained for Figure 5.15 (c) and (d), the lifetime of Backbone tree becomes extended but not as much as the PEGASIS chain or BFS tree. Yet in most of the cases ($r < 26$), the Backbone tree remains superior to the other two. Combining the above discussion and the results

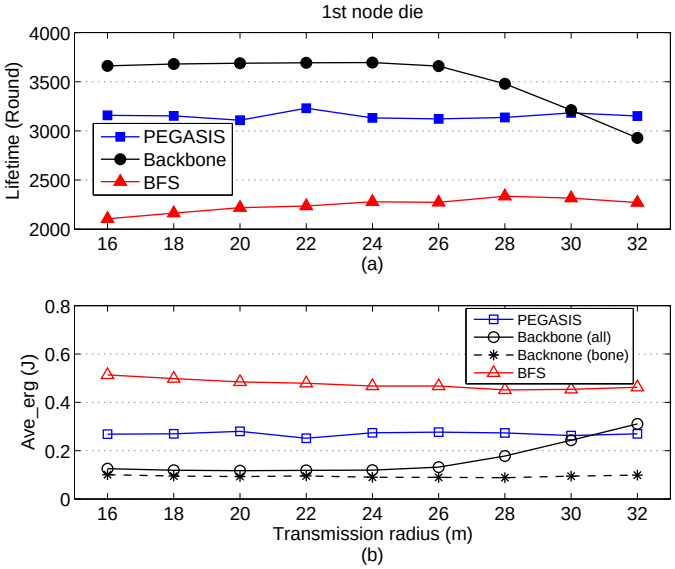


Figure 5.16: Lifetime and corresponding residual energy comparison of three algorithms under different transmission radii when the first node dies ($n = 200$).

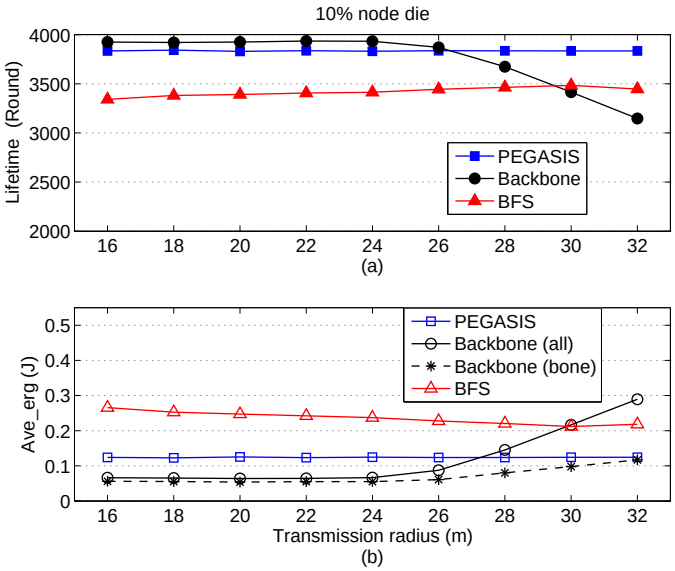


Figure 5.17: Lifetime and corresponding residual energy comparison of three algorithms under different transmission radii when 10% of nodes die.

in Figure 5.15, 5.16, and 5.17, we conjecture there exists a local extremum for the Backbone tree network lifetime under some certain neighboring number nodes. In the network setting of Figure 5.15 (a), (b) and Figure 5.16, we calculate that the maximum neighboring node is around 50. This extremum gives a clue to the network's node saturation for obtaining the best performance of the Backbone tree.

5.6 Summary

A data communication framework for the logical cluster in wireless sensor networks is developed in this chapter. The two components of the framework cooperatively solve the typical problems taking the data communication in logical clustered sensor networks into account, which includes the overlapping of transmission areas and the routing in diversely shaped clusters. The light-weight and self-organizing working/sleeping schedule demonstrates member identification, interference elimination and energy conservation's three fold functions. The Backbone tree self-grows along the diverse shape of the logical cluster and builds a routing plan for each cluster member.

Chapter 6

Conclusion

6.1 Conclusion

Inspired from the biological synchronicity phenomena, this thesis proposes a new logical clustering method based on the synchronicity and desynchronicity characteristics of node oscillators. Different from the existing clustering techniques, the GCC method imitates the mutual acting in neuron dynamic system to self-autonomously achieve a global clustering via pulse-coupled communications. The used clustering criteria can generally come from any node-related data or properties, and its selection can be influenced by the application level requirements. This broad choice of clustering criteria indicates the generality of the method that results in new cluster formations not otherwise existing.

The intra/inter data communication on logical clustered network are achieved through the proposed framework. Using the inherent desynchronicity and synchronicity properties, the quasi-TDMA scheme and Backbone tree topology exhibit advantages of light weight, interference free, and load balance, and more. However, they are not exclusive, and the two components of the framework can be replaced by any alternative similar functional solutions. Furthermore, the synchronicity and desynchronicity primitive in the logical clustered networks can provide more sup-

porting services to increase the performance of networks.

The function of simple sensor nodes self-organizing into groups according to application requirements makes the whole system demonstrate some elementary intelligence. This function is similar with the human brain and optic neuron system working mechanism, and it is the foundation of further advanced system intelligence and cognition. We believe that new perceptions about clustering are enabled and the GCC method demonstrates broadly applicable potential in new areas, such as the image segmentation, object capturing, data classifying, information procession, etc.

6.2 Future Work

During the course of research, observations about the performance of GCC are made, some of which have been pointed out in this thesis. These observations include the system converging speed, the fine tuning of the control parameters, and alternative pulse signal techniques. Further investigation on the following items are needed: the converging mechanism of DIPCO model in multiple node system, the influence of the pulse signal detection technique, and wireless dynamics on the clustering results, etc.

In Chapter 5, the basic performance of the data communication on the framework were analyzed, such as the network response speed, energy balancing, and the lifetime. However, further analysis on the packet loss, network throughput, security and resilience of the framework have not yet been explored. All of the above directions encourage our future work.

Appendix A: Convergence Proof on Delayed Inhibitory Pulse-Coupled model

A.1 Mathematic Constrains in DIPCO

Here we use the same oscillating function form as in Section 3.2 that $x_i(t) = f(\phi_i(t))$ is monotonically increasing and concave down ($f' > 0$, $f'' < 0$), in which $\phi \in [0, 1]$, $f(\phi) \in [0, 1]$ and period $T = 1$. F_- and F_+ denote the phase after excitatory and inhibitory coupling:

$$\begin{aligned} F_+(\phi) &= MIN(1, f^{-1}(f(\phi) + \epsilon)) \\ F_-(\phi) &= MAX(0, f^{-1}(f(\phi) - \epsilon)) \end{aligned}$$

Before treating a pair of these oscillators in a mathematical analysis, some simple properties of the functions f and F_- are needed in the next paragraph:

A1: $f(\phi) \geq \phi$, for $0 \leq \phi \leq 1$.

A2: $f(\phi_1) + f(\phi_2) > f(\phi_1 + \phi_2)$, for $0 < \phi_1 + \phi_2 \leq 1$.

$$
\begin{aligned}
f(\phi_1+\phi_2) &= f(\phi_1)+\Delta y_{\phi_1} \\
\frac{dy}{d\phi} &= f'(\phi) \\
\text{so} \qquad dy &= f'(\phi)d\phi \\
\Delta y_{\phi_1} &= \int_{\phi_1}^{\phi_1+\phi_2} f'(\phi)d\phi \\
\text{While} \quad f(\phi_2) &= \int_0^{\phi_2} f'(\phi)d\phi
\end{aligned}
$$

Because of $f' > 0$ and $f'' < 0$, the $f'(\phi)$ in interzone $(0, \phi_2)$ is bigger than in $(\phi_1, \phi_1+\phi_2)$. So

$$
\begin{aligned}
f(\phi_2) &> \Delta y_{\phi_1} \\
f(\phi_1)+f(\phi_2) &> f(\phi_1+\phi_2)
\end{aligned}
$$

With conclusion of rule A2, the next A3 A4 rules can be inferred.

A3: $c \cdot f(\phi) > f(c \cdot \phi)$, $c > 0$ is any constant.

A4: $f(\phi_1) - f(\phi_2) < f(\phi_1 - \phi_2)$ for $\phi_1 > \phi_2 > 0$.

A5: $F_-(\phi) < \phi$.

A6: $F_-(\phi_1) - F_-(\phi_2) < \phi_1 - \phi_2$, for $\phi_1 > \phi_2 > 0$

A7: $F'_-(\phi) > 0$, that means if $\phi_1 > \phi_2$, $F_-(\phi_1) > F_-(\phi_2)$.

$$F_-(\phi) = f^{-1}(f(\phi) - \epsilon)$$

$$\begin{aligned} F'_-(\phi) &= f^{-1\prime}(f(\phi) - \epsilon) f'(\phi) \\ &= \frac{f^{-1\prime}(f(\phi) - \epsilon)}{f^{-1\prime}(f(\phi))} \end{aligned}$$

For $f' > 0$ and $f^{-1\prime\prime} > 0$, so $f^{-1\prime}(f(\phi) - \epsilon) < f^{-1\prime}(f(\phi))$,

$$0 < F'_-(\phi) < 1. \tag{A.1}$$

A.2 Phase District

In [24], a node reacts to every incoming signal except the one emitted by itself. It is theoretically reasonable that nodes do not react to themselves. In reality, however, the node cannot tell the pulse source because all incoming signals are the same simple short pulses. Some systems set refractory periods to avoid the self-reaction, but when the node number is more than two, not only self-reaction, but all the reactions that happens within refractory period will be restrained. With the non-self-reaction rule, all nodes adjust their phase values except the firing one, thus changing the node firing order and ending the system in to the *Marginal Stable state*. Conversely, the refractory period is designed to avoid the chain-reacting caused by continuous firing, but the inhibitory coupling always postpones the next firing event, which automatically avoids the continuous firing. The refractory period is not longer necessary. Thus nodes here can react to all the incoming signals including the ones emitted by themselves, and this result is called the **All-Reaction** scheme.

For the consideration of delay, the original IF curve extends a τ length after the firing point. The reaction will not happen instantaneously after the firing as ideally assumed, instead after τ

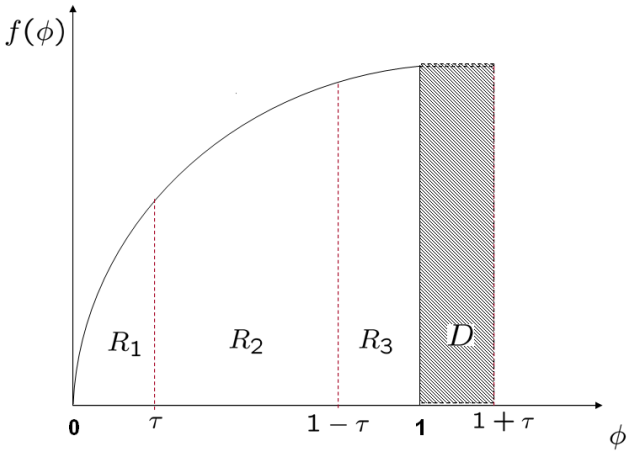


Figure A.1: District zones in one oscillating period .

time. Using the phase scale to measure the delayed time, τ length before and after the firing point become the critical districts. Assume the delay time is less than half of the period $\tau < \frac{1}{2}T$, thus, $\tau < 1 - \tau$. The critical districts divide the phase period T into three regions: $R_1 : (1, \tau)$, $R_2 : (\tau, 1 - \tau)$, $R_3 : (1 - \tau, 1)$. Figure A.1 shows the district division and delay region.

This part also contributes two concepts to describe the dynamics of the delay pulse inhibitive coupling oscillator system: the *'floating signal number'* and *'system state'*. When there are two nodes in the system, I_{ij} represents the system state. $i \in \{1, 2, 3\}$ is the district number of one node phase locates at when the other node phase is zero. $j \in \{0, 1, 2\}$ means how many firing signals are floating in the air waiting for reaction. Two nodes' delayed inhibitory coupling system can start from three starting states: I_{10}, I_{20}, I_{30}.

A.3 Two Nodes' Finite State Machine

This section analyzes the converging situation of two nodes initiated from tree starting states. For convenience and comparison of two reaction schemes, use the formula format similar to [24]. The tabular expressions show every step of phase changing in the first round, which helps to predict the system's final state. In the end, it figures out the whole system's finite state machine.

When reacting to the pulses, different coupling strengths ϵ will cause different results. Therefore, for the same starting state, according to the value domain of ϵ, there would appear several transfer cases. In following text, the symbol '**Ina.b.c**' means the inhibitive coupling system starts from region '**a**', case '**b**', and subcase '**c**' situation.

First, assuming two nodes A, B with initial phases 0 and ϕ, $h(\phi)$ is the phase difference value after one round's pulse reaction.

Starting from district R_1

When $\phi \in I_{10}$, node B first fire and reset, A moved to $1-\phi$. Next A need ϕ time to get fired, and B's pulse will be reacted in τ time (Figure A.2). Since node B Starts from region R_1, $\phi < \tau$ and A will get fire first. Then there will be two signals in air waiting for reaction. The coupling strength boundary is set by the resulted phase values (a_1, b_1) after the first reaction, which may be both bigger than 0 (In1.1, $a_1 > 0$), only one bigger than 0 (In1.2, $a_1 \leq 0$), or both less than 0 (In1.3, $a_1 \leq 0$, $b_1 \leq 0$). Further offshoots of every case are decided by the phase value after the second reaction.

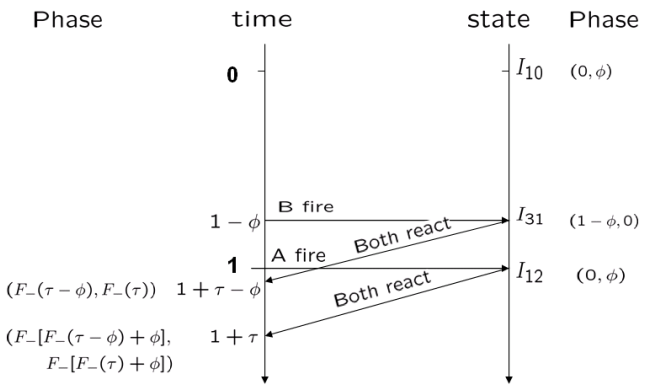


Figure A.2: Timeline and state change when phase ϕ_B started district R_1.

In1.1: $\epsilon < f(\tau - \phi), \phi \in I_{10}$

time	ϕ_A	ϕ_B	signal	*state*
0	0	ϕ	0	I_{10}
$1-\phi$	$1-\phi$	$1 \to 0$	1	I_{31}
1	$1 \to 0$	ϕ	2	I_{12}
$1+\tau-\phi$	$a_0 = \tau - \phi \to a_1 = F_-(a_0) > 0$	$b_0 = \tau \to b_1 = F_-(b_0)$	1	
$1+\tau$	$a_1 + \phi \to a_2 = F_-[a_1 + \phi]$	$b_1 + \phi \to b_2 = F_-[b_1 + \phi]$	0	

Because $a_0 < b_0$ and $F_-(\phi)$ is monotonic increasing (rule A7), then $0 < a_1 < b_1$, $a_1 + \phi < b_1 + \phi$ and $a_2 < b_2$. $\epsilon < f(\tau - \phi) < f(\tau)$ makes $0 < a_1 < b_1$, under this circumstance:

In1.1.1 when $a_2 > 0$

$$\begin{aligned} h_1(\phi) &= |b_2 - a_2| \\ &= |b_1 - \Delta b_1 - a_1 + \Delta a_1| \\ &< |b_1 - a_1| \\ &= |b_0 - \Delta b_0 - a_0 + \Delta a_0| \\ &< |b_0 - a_0| \\ &= \phi \end{aligned}$$

The two inequations are supported by the fact that coupling happened in later positions always causes larger phase jump amounts than the former one when the oscillating function is concave down [55]. Here $a_0 < b_0$ and $a_1 < b_1$, then $\Delta a_0 < \Delta b_0$, $\Delta a_1 < \Delta b_1$, so $h(\phi) < \phi$.

In 1.1.2 when $a_2 \leq 0$ but $b_2 > 0$

$$\begin{aligned} h_2(\phi) &= |b_2 - 0| \\ &\leq |b_2 - a_2| \\ &< \phi \end{aligned}$$

In1.1.3 when $a_2 < b_2 \leq 0$

$$h_3(\phi) = |0 - 0| = 0$$

Combine the above three cases, $h(\phi) < \phi$ or $h(\phi) = 0$; the nodes phase difference either shrink or directly decrease to zero after one round reaction. And with the decreased ϕ, ϵ will always meet the requirement $\epsilon < f(\tau - \phi)$ in the future rounds. Thus, the phase difference will

continue decreasing until to 0, which means the node phases get synchronized.

In1.2: $f(\tau-\phi) \le \epsilon < f(\tau), \phi \in I_{10}$

$time$	ϕ_A	ϕ_B	signal	$state$
1	0	ϕ	2	I_{12}
$1+\tau-\phi$	$\tau-\phi \to F_-(\tau-\phi)=0$	$\tau \to F_-(\tau)$	1	I_{11}
$1+\tau$	$\phi \to F_-(\phi)$	$F_-(\tau)+\phi \to F_-(F_-(\tau)+\phi)$	0	

For different ϵ and initial ϕ values, there will appear three resulting states:

In1.2.1 when $F_-(\phi) > 0$:

$$\begin{aligned} h_1(\phi) &= |F_-(F_-(\tau)+\phi) - F_-(\phi)| \\ &< |F_-(\tau)+\phi-\phi| \qquad \text{(A6)} \\ &= |F_-(\tau)| \\ &= f^{-1}(f(\tau)-\epsilon) \\ &\le f^{-1}(f(\tau)-f(\tau-\phi)) \\ &< f^{-1}(f(\tau-\tau+\phi)) \qquad \text{(A4)} \\ &= \phi \end{aligned}$$

In1.2.2 when $F_-(\phi) \le 0$, $F_-(F_-(\tau)+\phi) > 0$:

$$\begin{aligned} h_2(\phi) &= |F_-(F_-(\tau)+\phi) - 0| \\ &< |F_-(F_-(\tau)+\phi) - F_-(\phi)| \\ &< \phi \end{aligned}$$

In1.2.3 when $F_-(\phi) < F_-(F_-(\tau)+\phi) \le 0$:

$$h_3(\phi) = |0-0| = 0$$

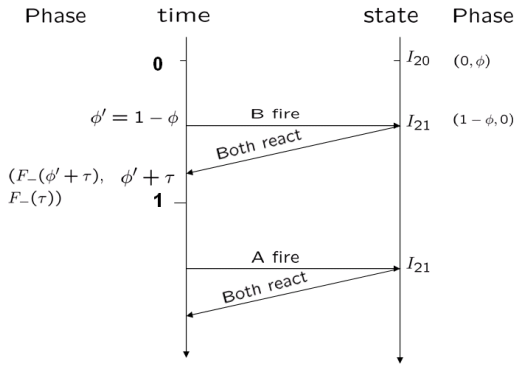


Figure A.3: Timeline and state change when phase ϕ_B started district R_2.

Same with case In1.1, $h(\phi) < \phi$ or $h(\phi) = 0$. The phase difference either shrinks or gets synchronized directly. But as the phase difference decreases in every round, the size of domain $f(\tau - \phi) \leq \epsilon < f(\tau)$ shrinks too. ϵ will finally underflow the boundary and jump into case In1.1.

In1.3: $\epsilon \geq f(\tau), \phi \in I_{10}$

$time$	ϕ_A	ϕ_B	signal	$state$
1	0	ϕ	2	I_{12}
$1 + \tau - \phi$	$\tau - \phi \rightarrow F_-(\tau - \phi) = 0$	$\tau \rightarrow F_-(\tau) = 0$	1	

When ϵ is big enough, two nodes will get synchronized immediately after coupling once.

Integrate the three cases(In1.1-In1.3): any two nodes whose initial phase difference is less than the delay value $|\phi| < \tau$, under the all-reaction rules, will definitely get synchronized. In other words, all the nodes initiated from R_1 will end up with the synchronized state whether directly or after several rounds.

Starting from district R_2

When starting from state I_{20}, after B fires, there are two events waiting: A fires or they both react to B's pulse. In region R_2, $\phi > \tau$, the reaction happens before A's firing. The coupling strength boundary is set by the resulted phase value after the first reaction, which may be both bigger than 0 (In2.1), only one bigger than 0 (In2.2) or both less than 0 (In2.3).

In2.1: $\epsilon < f(\tau), \phi \in I_{20}$

$time$	ϕ_A	ϕ_B	signal	$state$
0	0	ϕ	0	I_{20}
$1-\phi$	$1-\phi=\phi'$	$1 \to 0$	1	I_{21}
$\phi'+\tau$	$\phi'+\tau \to F_-(\phi'+\tau)$	$\tau \to F_-(\tau) > 0$	0	

$$\begin{aligned} h(\phi) &= |F_-(\phi'+\tau) - F_-(\tau)| \\ &< |\phi'+\tau-\tau| \\ &= \phi' \\ &< 1-\tau \end{aligned}$$

In2.1.1: If $F_-(\phi'+\tau) - F_-(\tau) \leq \tau$, after once reaction, the phase difference become less than τ, the system state turn to case In1.

In2.1.2: If $F_-(\phi'+\tau) - F_-(\tau) > \tau$, then $\tau < h(\phi) < 1-\tau$. Next, after shifting and fire of A, the system state will return I_{21} again.

Check the relationship of $h(\phi)$ and ϕ, set $H(\phi) = h(\phi) - \phi$,

$$\begin{aligned} H(\tau) &= F_-(1-\tau+\tau) - F_-(\tau) - \tau \\ &= 1 - F_-(\tau) - \tau \\ &> 1 - \tau - \tau \\ &= 1 - 2\tau \\ &> 0 \end{aligned}$$

Region R_2 $(\tau, 1-\tau)$ is not empty, so $\tau < 0.5$.

$$\begin{aligned} H(1-\tau) &= F_-(1-1+\tau+\tau) - F_-(\tau) - 1 + \tau \\ &= F_-(2\tau) - F_-(\tau) - 1 + \tau \\ &< 2\tau - \tau - 1 + \tau \\ &= 2\tau - 1 \\ &< 0 \end{aligned}$$

Check the monotonicity of $H(\phi)$, for $H(\phi) = h(\phi) - \phi$, $h(\phi) = |F_-(\phi' + \tau) - F_-(\tau)|$, then

$$\begin{aligned} H'(\phi) &= h'(\phi) - 1 \\ &= -F'_-(1-\phi+\tau) - 1 \end{aligned}$$

Because of rule A7, $0 < F'_-(\phi) < 1$

$$-1 < h'(\phi) < 0 \tag{A.2}$$

$$-2 < H'(\phi) < -1 \tag{A.3}$$

$H(\phi)$ is monotonic decrease. Because $H(\tau) > 0$ and $H(1-\tau) < 0$, there must exist one and only one point ϕ_0 in non-empty domain $(\tau, 1-\tau)$ that makes $H(\phi_0) = 0$, that $h(\phi_0) = \phi_0$.

For the return function $R(\phi) = h(h(\phi))$, its monotonicity is checked by $R' = h'(h(\phi))h'(\phi)$, and

$$0 < R'(\phi) < 1. \tag{A.4}$$

So,

$$\begin{aligned} R(\phi) &> \phi \qquad if \quad \phi < \phi_0, \\ R(\phi) &< \phi \qquad if \quad \phi > \phi_0. \end{aligned} \tag{A.5}$$

This fixed zero point ϕ_0 is an attractor. The two nodes will finally converge to $(0, \phi_0)$ stable state and never get synchronized, which is called the *Anti-Phase* stable state [24].

In2.2: $f(\tau) \leq \epsilon < f(\tau + \phi')$, $\phi \in I_{20}$

$time$	ϕ_A	ϕ_B	signal	$state$
$1-\phi$	$1-\phi = \phi'$	$1 \to 0$	1	I_{21}
$\phi' + \tau$	$\phi' + \tau \to F_-(\phi' + \tau) > 0$	$\tau \to F_-(\tau) = 0$	0	

In2.2.1 when $f(\phi' + \tau) > \epsilon \geq f(\phi' + \tau) - f(\tau)$.

$F_-(\phi' + \tau) \leq \tau$, so $h(\phi) \leq \tau$. After once reaction, the phase difference become less than τ,

the system turn to case In1.

In2.2.2 when $f(\tau) \leq \epsilon < f(\tau + \phi') - f(\tau)$, $h(\phi) > \tau$.

$$\begin{aligned} h(\phi) &= |F_-(\phi' + \tau)| \\ &= |f^{-1}(f(\phi' + \tau) - \epsilon)| \\ &\leq |f^{-1}(f(\phi' + \tau) - f(\tau)) \\ &< |f^{-1}(f(\phi' + \tau - \tau)| \qquad \text{(A4)} \\ &= \phi' \end{aligned}$$

So $h(\phi) < \phi' < 1 - \tau$, $h(\phi)$ is still in region R_2. We use the same method as case In2.1 to check their converging trend:

First $h(\phi) = F_-(\phi' + \tau)$, $H(\phi) = h(\phi) - \phi < \phi' - \phi$

$$\begin{aligned} H(\tau) &= F_-(1 - \tau + \tau) - \tau \\ &= 1 - \tau \\ &> 0 \end{aligned}$$

$$\begin{aligned} H(1 - \tau) &< (1 - (1 - \tau)) - (1 - \tau) \\ &= \tau - 1 + \tau \\ &= 2\tau - 1 \\ &< 0 \end{aligned}$$

Same with case In2.1.2, $H(\phi)$ is monotonically decrease ($-1 < h'(\phi) < 0$ and $-2 < H'(\phi) < -1$). There exist the only one zero point as an attractor ($0 < R'(\phi) < 1$) in $(\tau, 1 - \tau)$.

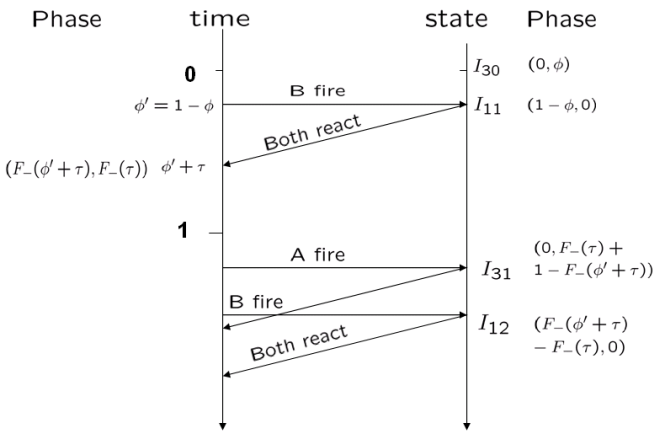


Figure A.4: Timeline and state change when phase ϕ_B started district R_3.

The system ends up with the *Anti-phase* stable state as well.

In2.3: $\epsilon \geq f(\phi' + \tau))$, $\phi \in I_{20}$

$time$	ϕ_A	ϕ_B	signal	$state$
ϕ'	$1 - \phi$	$1 \to 0$	1	I_{21}
$\tau + \phi'$	$\phi' + \tau \to F_-(\phi' + \tau) = 0$	$\tau \to F_-(\tau) = 0$	0	

$h(\phi) = |0 - 0|$, the nodes synchronized directly.

Under some value of coupling strength, two nodes who initiate from state I_{20} will not synchronize but converge to the *Anti-phase* stable state.

Starting from district R_2

In3.1: $\epsilon < f(\tau), \phi \in I_{30}$

$time$	ϕ_A	ϕ_B	signal	$state$
0	0	ϕ	0	I_{30}
$1-\phi$	$1-\phi=\phi'$	$1 \to 0$	1	I_{11}
$\phi'+\tau$	$\phi'+\tau \to F_-(\phi'+\tau)>0$	$\tau \to F_-(\tau)>0$	0	
...	$1 \to 0$	$F_-(\tau)+1-F_-(\phi'+\tau)$	1	I_{31}
...	$F_-(\phi'+\tau)-F_-(\tau)$	$1 \to 0$	2	I_{12}

When initial phase ϕ is in district R_3, $1-\tau < \phi < 1$. After the first reaction, if two phases are both still bigger than zero,

$$\begin{aligned} h_1(\phi) &= |F_-(\phi'+\tau)-F_-(\tau)| \\ &< \phi'+\tau-\tau \\ &= \phi' \\ &< \tau \end{aligned}$$

The phase difference becomes less than τ, and the system in future rounds will turn to case In1 followed by the state I_{31} and I_{12} (figure A.4).

In3.2: $f(\tau) \le \epsilon < f(\phi'+\tau), \phi \in I_{30}$.

$time$	ϕ_A	ϕ_B	signal	$state$
$1-\phi$	$1-\phi=\phi'$	$1 \to 0$	1	I_{11}
$\phi'+\tau$	$\phi'+\tau \to F_-(\phi'+\tau)>0$	$\tau \to F_-(\tau)=0$	0	

$$\begin{aligned} h_2(\phi) &= F_-(\phi' + \tau) \\ &\leq f^{-1}[f(\phi' + \tau) - f(\tau)] \\ &< f^{-1}[f(\tau + \tau) - f(\tau)] \\ &< f^{-1}[f(\tau + \tau - \tau)] \\ &= \tau \end{aligned}$$

In3.3: $\epsilon \geq f(\phi' + \tau)$, $\phi \in I_{30}$.

$time$	ϕ_A	ϕ_B	signal	$state$
$1 - \phi$	$1 - \phi = \phi'$	$1 \rightarrow 0$	1	I_{11}
$\phi' + \tau$	$\phi' + \tau \rightarrow F_-(\phi' + \tau) \leq 0$	$\tau \rightarrow F_-(\tau) \leq 0$	0	

$h_3(\phi) = |0 - 0| = 0$.

Nodes initiated from region R_3, after one reaction, will all jump into region R_1. With the conclusion of case In1, all the nodes will finally get synchronized, no matter how strong the coupling strength is.

Figure A.5 shows the finite state machine of two nodes with delayed inhibitory coupling. There are three starting states, several intermediate states and two ending stable states: *Synchronization* and *Anti-phase*. Due to the all-reaction scheme, there are no *marginal stable* states as in reference [24], which use the non-self-reaction scheme. The figure shows that one can arrive at the synchronization ending state from every starting state either directly (Arc 3,1,6) or after finite rounds of intermediate state transferring (Arc 2,7). However the Anti-Phase state only can only be reached by starting from I_{20} (Arc 4). The state transferring arrows in figure A.5 mostly are single direction except for two cases: Arc 2 is bidirectional, and 5 is a circle. The to-and-fro transfers (Arc 2) between I_{31} and I_{12} represent the steps to decrease the phase difference. The convergence

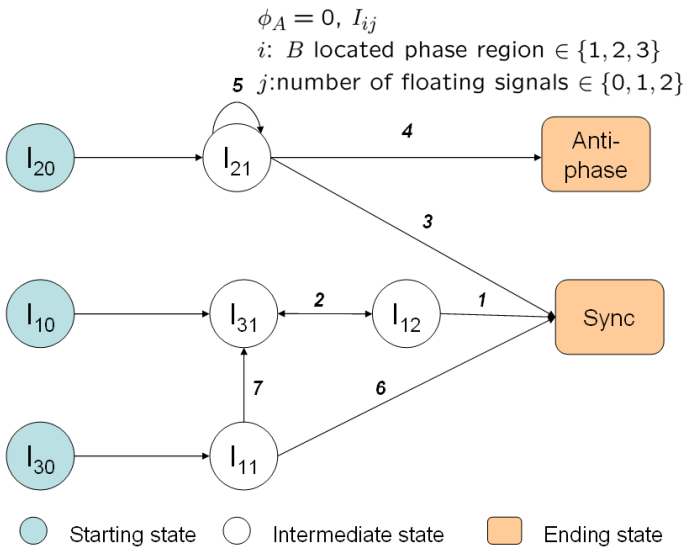


Figure A.5: Finite state machine of two nodes inhibitive coupling.

is proven by case In1.1 and In1.2, so this iteration transferring number is finite. Cycle 5 shows the system may converge to the attractor point in region R_2 and ends up with *Anti-phase* state, case In2.1 and In2.2 prove the convergence.

A.4 Heuristic Conjecture of Three and Multiple nodes Situation

In n nodes system, each pair of nodes phase obey the same rule as the two nodes situation. The other $n-2$ nodes can be regarded as one unit acting as the third 'node' in system. This special node can have arbitrary coupling strength and phase value. This part will discuss the three nodes system converging procedure as a bridge to investigate the similar procedure of n nodes system.

First of all, the research of two nodes converging procedure reveals that when two nodes phase difference is kept fewer than delay times τ, they will finally converged into one group; but when

their phase distance is bigger than τ, they are expelled to some distance. Thus, the τ is a critical length to decide two node converge or not.

Among three nodes, there can pick up three pairs of nodes, so there are three phase distances. Here we set a three bits number, in which each bit denotes one phase distance between two neighboring nodes. If the phase distance is less than τ, the corresponding bit is set as 0, otherwise set as 1. If two continued distances are both less than τ (two continuous 0), then add one more bit to denote whether or not the sum of two previous distances is still less than τ. Thus, all possibilities of the three nodes' phase distributions can be represented by a three or four-bit binary number.

Extending this representation rule, a n nodes system can be represented using at most $n+(n-1)(n-2)/2$ bits. Here the first item means the apparent n phase distances, while $(n-1)(n-2)/2$ is the most possible added bits number. For examples, for 2 nodes system, there are only need two bits to represent the two nodes' bidirectional phase distances and no necessary extra bits. While for the 4-nodes system, besides 4 basic bit, there may need to be at most 3 additional digits to represent the other non-neighboring but closed node pair distances. With this binary number, we can denote every possible system state. In most cases, the total probable system states number is much less than $2^{n+(n-1)(n-2)/2}$ because the extra bits added mean the more assured zero bits there continue to be.

Figure A.6 shows the three node state transferring situation. From it, the similar conclusion can be drawn that under the third node's influence, the two nodes phase difference may decrease or increase somewhat, but if two nodes' phase difference is less than the critical length τ, they are high probably that they get converged; and the two whose phase difference is always bigger than τ will be repelled in most cases.

As set in Section A.1, the nodes follow the monotonic increase, concave-down oscillating function in domain $[0, 1]$, so all the phases that are back jumped less than 0 will be regarded as 0. These nodes are absorbed in the zero point, so as to be considered synchronized.

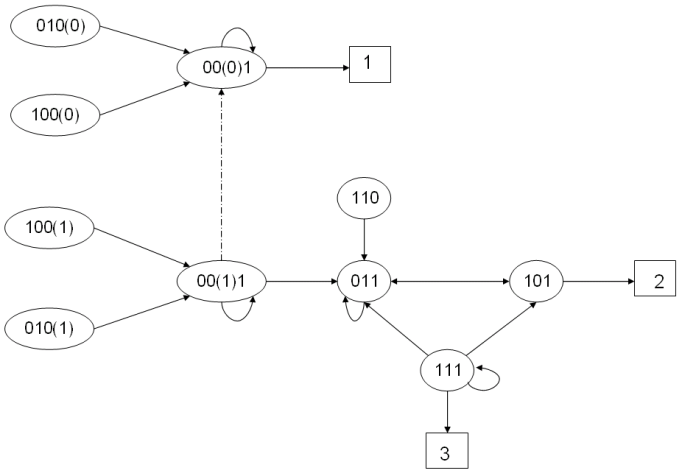


Figure A.6: Ordered three nodes' state transferring.

Refer the two nodes converging procedure and the mechanism of absorbing, the n nodes system converging procedure can be conjectured as follows:

1. Every node can act as an absorber when it is at zero point as long as the coupling strength is strong enough or the neighboring nodes phase are close enough.

2. After absorbing other nodes, the number of group member increases. Accordingly, the group coupling strength and the absorbing radius increases too, which makes the group absorb new nodes more easily in the next round.

3. However, the absorbing radius of a group is at most as big as τ. The research of the two and three nodes converging procedure reveals that the τ phase length before and after absorber are two critical districts for synchronizing.

4. So under the synthetic work of both two oscillators and inhibitory coupling, the system nodes in range $[0, 1]$ will finally converge to several clusters. There is no fixed cluster center but the cluster radius is around τ length and the cluster number is proportional to $1/2\tau$.

Appendix B: List of Publications

Journal Publications

1. **Yu Niu**, Brian. J. d'Auriol, and Sungyoung Lee, General criteria-based clustering method for multi-node computing system, *The Journal of Supercomputing*, ISSN:0920-8542, (Online first) 2011. doi:10.1007/s11227-011-0642-0 **(SCI)**

2. **Yu Niu**, Jin Wang, Soungyoung Lee and Young-Koo Lee, Backbone Tree based Data Gathering protocol for Wireless Sensor Networks, *AISS: Advances in Information Sciences and Service Sciences*, ISSN: 1976-3700, Vol. 3, No. 8, pp. 351-360, 2011. **(EI)**

3. Jin Wang, Jeong-Uk Kim, Lei Shu, **Yu Niu** and Sungyoung Lee, A Distance-Based Energy Aware Routing Algorithm for Wireless Sensor Networks, *Sensors*, 2010. ISSN: 1424-8220, Vol. 10, No. 10, pp. 9493-9511. doi:10.3390/s101009493. **(SCIE)**

Conference Publications

1. **Yu Niu**, Brian. J. d'Auriol, and Sungyoung Lee, A Novel Tree based Data Gathering Protocol for Wireless Sensor Networks. In: *Proceedings of The 7th International Conference*

on Networked Computing and Advanced Information Management (NCM 2011), Gyeongju, Korea, June 21-23, 2011.

2. **Yu Niu**, Brian. J. d'Auriol, and Sungyoung Lee, General Criteria-based clustering method for sensor network. In: *Proceedings of The 2011 International Conference on Network Computing and Information Security (NCIS'11) (IEEE)*, Guilin, China, May 14-15, 2011.

3. **Yu Niu**, Brian. J. d'Auriol, Young-Koo Lee, and Sungyoung Lee, An Analysis Method for Dynamical System. In: *Proceedings of The 32th KIPS Winter Conference*, Konkuk University, Seoul, Korea, Nov 13-14, 2009.

4. Brian. J. d'Auriol, **Yu Niu**, Sungyoung Lee, and Young-Koo Lee, The Plasma Free Space Optical Model for Ubiquitous Systems. In: *Proceedings of The 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC 2009)*, pp. 446-455, January 15-16, 2009, Sungkyunkwan University, Suwon, Korea, ACM Press.

5. **Yu Niu**, Brian. J. d'Auriol, Xiaoling Wu, Jin Wang, Jinsung Cho, and Sungyoung Lee, Selective Pulse Coupling Synchronicity for Sensor Network. In: *Proceedings of The Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008)*, Cap Esterel, France, August 25-31, 2008.

6. Wang Jin, **Yu Niu**, Jinsung Cho, and Sungyoung Lee, Analysis of energy consumption in direct transmission and multi-hop transmission for wireless sensor networks. In: *Proceedings of The Third International Conference on Signal-Image Technology & Internet-Based Systems (SITIS 2007)*, Shaghai, China , December 16-19, 2007.

7. Brian J. d'Auriol, Jie Yang, Xiaoling Wu, Hui Xu, **Yu Niu**, Jin Wang, Riaz Ahmed Shaikh, Min Meng, Sungyoung Lee and Young-Koo Lee, A Research Framework Model to Guide

Both Broad and Focused Research into Ubiquitous Sensor Networks. In: *Proceedings of The 2007 International Conference on Wireless Networks (ICWN'07)*, Las Vegas, Nevada, USA., June 25-28, 2007.

8. **Yu Niu**, Brian. J. d'Auriol, Young-Koo Lee, and Sungyoung Lee, A Fast Converging Pulse Coupling Oscillator Synchronicity Model. In: *Proceedings of The 27th KIPS Spring Conference*, Kyungwon University, Seoul, Korea, May 11-12, 2007, pages 860-861.

9. Xiaoling Wu, **Yu Niu**, Lei Shu, Jinsung Cho, Young-Koo Lee and Sungyoung Lee, Relay Shift Based Self-Deployment for Mobility Limited Sensor Networks. In: *Proceedings of The 3rd International Conference on Ubiquitous Intelligence and Computing (UIC-06)*, pp. 556-564, Wuhan and Three Gorges, China, September 3-6, 2006.

10. Hui Xu, Manwoo Jeon, Jinsung Cho, **Yu Niu** and Sungyoung Lee, Effect of Realistic Physical Layer on Energy Efficient Broadcast Protocols for Wireless Ad Hoc Networks. In: *Proceedings of 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN-06)*, pp.327-339, St.Petersburg, Russia, May 29 - June 2, 2006.

11. Hui Xu, Manwoo Jeon, Shu Lei, **Yu Niu**, Jinsung Cho and Sungyoung Lee, Impact of Practical Models on Power Aware Broadcast Protocols for Wireless Ad Hoc and SensorNetworks. In: *Proceedings of The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS-06)*, pp. 133-140, Gyeongju, Korea, April 27-28, 2006.

References

[1] IEEE 802.11g-2003: Further higher data rate extension in the 2.4 GHz band (PDF). Technical report, IEEE, Oct 2003. Retrieved 2007-09-24.

[2] International system of units (SI). Technical report, Bureau International des Poids et Measures, 2006. 8th edition.

[3] 802.15.4a-2007 - IEEE standard for local and metropolitan area networks - specific requirements - part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs): Amendment 1: Add alternate phys. Technical report, IEEE, Aug 2007. Amendment to IEEE Std 802.15.4-2006.

[4] I. Aihara, H. Kitahata, K. Yoshikawa, and K. Aihara. Mathematical modeling of frog's calling behavior and its possible applicatio n to artificial life and robotics. *Artificial Life and Robotics*, 12(1-2):29–32.

[5] I. Aihara and K. Tsumoto. Mathematical modeling of frog's calling behavior and its possible applicatio n to artificial life and robotics. *Mathematical Biosciences*, 214(1-2):6–10.

[6] A. D. Amis, R. Prakash, T. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings of Nineteenth Annual Joint Conferenceof the IEEE Computer and Communications Societies. INFOCOM 2000*, volume 1, pages 32–41, Tel-Aviv, Israel, Mar 2000.

[7] H. Arslan, Z. N. Chen, and M.-G. D. Benedetto. *Ultra WideBand Wireless Communication*, chapter 14, pages 318–339. Wiley-InterScience, Hoboken, New Jersey, 2006.

[8] P. Ashwin and M. Timme. Unstable attractors: existence and robustness in networks of oscillators with delayed pulse coupling. *Nonlinearity*, 18(5):2035–2060, 2005.

[9] P. Ashwin and M. Timme. When instability makes sense. *Nature*, 436(7):36–37, 2005.

[10] M.-G. D. Benedetto, L. D. Nardis, M. Junk, and G. Giancola. (UWB)2: Uncoordinated, wireless, baseborn medium access for uwb communication networks. *Mobile Networks and Applications*, 10(5):663–674, Oct 2005.

[11] H. W. Broer, K. Efstathiou, and E. Subramanian. Heteroclinic cycles between unstable attractors. *Nonlinearity*, 21(6):1385–1410, Jun 2008.

[12] H. W. Broer, K. Efstathiou, and E. Subramanian. Robustness of unstable attractors in arbitrarily sized pulse-coupled systems with delay. *Nonlinearity*, 21(1):13–49, Jan 2008.

[13] J. Buck. Synchronous rhythmic flashing of fireflies ii. *The quarterly Review of Biology*, 63(3):265–289, September 1988.

[14] S. R. Campbell, D. L. Wang, and C. Jayaprakash. Synchrony and desynchrony in integrate-and-fire oscillators. *Neural Computation*, 11(7):1595–1619, 1999.

[15] H. Chan and A. Perrig. ACE: An emergent algorithm for highly uniform cluster formation. In *Proceedings of First European Workshop Sensor Networks (EWSN)*, volume 2920, pages 154–171, Berlin, Germany, Jan 2004.

[16] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing.*, 5(2):193–204, Apr 2002.

[17] Chia-ChuChen. Threshold effects on synchronization of pulse-coupled oscillators. *Physical Review E.*, 49(4):2668–2672, 1994.

[18] B. J. dAuriol, Y. Niu, S. Y. Lee, and Y. K. Lee. The plasma free space optical model for ubiquitous systems. In *Proceedings of The 3rd International Conference on Ubiquitous Information Management and Communication. (ICUIMC-09)*, pages 446–455, Sungkyunkwan University, Suwon, Korea, Jan 2009. ACM Press.

[19] J. Degesys and R. Nagpal. Towards desynchronization of multi-hop topologies. In *Proceedings of the second IEEE international conference on self-adaptive and self-organizing systems. (SASO)*, pages 129–138, Venice, Italy, Oct 2008. IEEE Computer Society.

[20] J. Degesys, I. Rose, A. Patel, and R. Nagpal. Desync: Self-organizing desynchronization and TDMA on wireless networks. In *Proceedings of the Sixth International Conference on Information Processing in Sensor Networks (IPSN)*, pages 11–20, Cambridge, MA, USA, Apr 2007. ACM.

[21] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, pages 147–163, Boston, Massachusetts, USA, Dec 2002. USENIX Association.

[22] M. Ercsey-Ravasz, Z. Sarkzi, Z. Neda, A. Tunyagi, and I. Burda. Collective behavior of electronic fireflies. *The European Physical Journal B*, 65(2):271–277.

[23] U. Ernst, K. Pawelzik, and T. Geisel. Synchronization induced by temporal delays in pulse-coupled oscillators. *Phys. Rev. Lett.*, 74(9):1570–1573, Feb 1995.

[24] U. Ernst, K. Pawelzik, and T. Geisel. Delay-induced multistable synchronization of biological oscillators. *Phys. Rev. E*, 57(2):2150–2162, Feb 1998.

[25] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. In *Proc. 1st ACM Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 138–149, Los Angeles, California, USA, Nov 2003. ACM Press.

[26] S. Genaud, P. Gancarski, G. Latu, A. Blansch, C. Rattanapoka, and D. Vouriot. Exploitation of a parallel clustering algorithm on commodity hardware with P2P-MPI. *The Journal of Supercomputing*, 43(1):21–41, 2008.

[27] W. Gerstner. *The spike response model*, volume 4, chapter 12, pages 469–516. Elsevier Science, 2001.

[28] A. K. Ghosh, S. Kunta, P. Verma, , and R. C. Huck. Free-space optics based sensor network design using angle-diversity photodiode arrays. In *Proceedings of SPIE*, volume 7814, 78140U, San Diego, CA, USA, Aug 2010.

[29] B. Girau and C. Torres-Huitzil. Fpga implementation of an integrate-and-fire legion model for image segmentation. In *European Symposium on Artificial Neural Networks Bruges, ESANN'2006*, Belgium, Apr 2006.

[30] X. Guardiola, A. Diaz-Guilera, M. Llas, and C. J. Perez. Synchronization, diveristy, and topology of networks of integrate and fire oscillators. *Physical Review E*, 62(4):5565–5570, Oct 2000.

[31] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless sensor networks. In *Proceedings of the 33th Hawaii International Conference on System Sciences*, volume 2, page 10, Hawaii, USA, Jan 2000.

[32] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on wireless communications*, 1(4):660–670, Oct 2002.

[33] A. V. M. Herz and J. Hopfield. Earthquake cycles and neural reverberations: Collective oscillations in systems with pulse-coupled threshold elements. *Physical Review Letters*, 75(6):1222–1225, 1995.

[34] Y. W. Hong and A. Scaglione. Time synchronization and reach-back communications with pulse-coupled oscillators for UWB wireless ad hoc networks. In *Proceeding of IEEE Conference on Ultra Wideband Systems and Technologies (UWBST)*, Reston, VA, USA, Nov 2003.

[35] Y. W. Hong and A. Scaglione. Time synchronization with pulse-coupled oscillators for UWB wireless ad hoc networks. In *Proceedings of IEEE Conference on Ultra Wideband Systems and Technologies*, pages 190–194, Reston, Virginia, USA, Nov 2003.

[36] Y. W. Hong and A. Scaglione. A scalable synchronization protocol for large scale sensor networks and its applications. *IEEE Journal on Selected Areas in Communications*, 23(5):1085–1099, May 2005.

[37] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MOBICOM)*, Philadelphia, PA, USA, Sept 2004.

[38] P. Kumarawadu, D. J. Dechene, M. Luccini, and A. Sauer. Algorithms for node clustering in wireless sensor networks: A survey. In *4th International Conference on Information and Automation for Sustainability, 2008. (ICIAFS 2008).*, pages 295–300, Sri Lanka, Dec 2008.

[39] B. Kusy and M. Maroti. Flooding time synchronization in wireless sensor networks. In *Proceedings of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 39–49, Baltimore,Mariland, USA, Nov 2004. ACM Press.

[40] S. Lee and R. Listerf. Experiments in the dynamics of phase coupled oscillators when applied to graph colouring. In *Thirty-First Australiasian Computer Science Conference (ACSC2008)*, New South Wales, Australia, Jan 2008.

[41] N. Li, J. C. Hou, and L. Sha. Design and analysis of an MST-based topology control algorithm. *IEEE Transactions on Wireless Communications*, 4(3):1195–1206, May 2005.

[42] T. Lindblad and J. M. Kinser. *Image processiong using pulse coupled neural networks*. Springer-Verlag, 1998.

[43] S. Lindsey and C. S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Proceedings of IEEE Aerospace Conference, 2002*, volume 3, pages 1125–1130, Big Sky, Montana, USA, Mar 2002.

[44] X. Liu and D. L. Wang. Range image segmentation using a relaxation oscilatory network. *IEEE Transactions on Neural Networks*, 10(3):564–573, 1999.

[45] Y. Liu, Y. K. Kwok, and J. Z. Wang. On scheduling and clustering in hierarchical TH-PPM UWB wireless ad hoc networks. *The Journal of Supercomputing*, 46(1):58–83, 2008.

[46] D. Lucarelli and I. Wang. Decentralized synchronization protocols with nearest neighbor communication. In *Proceedings of 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 62–68, Baltimore,Mariland, USA, Nov 2004.

[47] N. Masuda and K. Aihara. Global and local synchrony of coupled neurons in small-world networks. *Biological Cybernetics*, 90(4):302–309, 2004.

[48] A. Mauroy and R. Sepulchre. Twofold impact of delayed feedback on coupled oscillators. *International Journal of Bifurcation and Chaos*, 7(17):2517–2530, Jul 2007.

[49] A. Mauroy and R. Sepulchre. Clustering behaviors in networks of integrate-and-fire oscillators. *Chaos*, 18(3):037122–037122–8, 2008. Special issue on Synchronization in Complex Networks.

[50] R. Mirollo and S. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662, Dec 1990.

[51] C. Muhlberger and R. Kolla. Extended desynchronization for multi-hop topologies. Technical Report 460, Institute of Information, University Wurzburg, Jul 2009.

[52] H. Nakano, A. Utani, A. Miyauchi, and H. Yamamoto. Synchronization-based data gathering scheme using chaotic pulse-coupled neural networks in wireless sensor networks. In *IEEE International Joint Conference on Neural Networks, 2008.*, pages 1115–1121, HongKong, Jun 2008. IEEE.

[53] Y. Niu, B. J. d'Auriol, and S. Y. Lee. A novel tree based data gathering protocol for wireless sensor networks. In *Proceedings of The 7th International Conference on Networked Computing and Advanced Information Management (NCM2011)*, pages 138–143, Gyeongju, Korea, Jun 2011. IEEE.

[54] Y. Niu, B. J. d'Auriol, Y. K. Lee, and S. Y. Lee. An analysis method for dynamical system. In *Proceedings of The 32th Korea Information Processing Society (KIPS) Fall Conference*, pages 45–P2–086, Seoul, Korea, Nov 2009.

[55] Y. Niu, B. J. d'Auriol, X. L. Wu, J. Wang, J. S. Cho, and S. Y. Lee. Selective pulse coupling synchronicity for sensor network. In *Second International Conference on Sensor Technologies and Applications 2008, (SENSORCOMM '08)*, pages 123–128, Cap Esterel, France, Aug 2008.

[56] A. Patel and J. Degesys. Desynchronization: The theory of self-organizing algorithms for round-robin scheduling. In *Proceedings of the first IEEE International Conference on self-adapting and self-organizing systems (SASO)*, pages 87–96, Cambridge. MA, USA., Jul 2007. IEEE Computer Society.

[57] C. Peskin. *Mathematical Aspects of Heart Physiology*, pages 268–278. Courant Institute of Mathematical Sciences, New York University, New York, 1975.

[58] S. Ping. Delay measurement time synchronization for wireless sensor networks. Technical report, Inter Research, Jun 2003.

[59] G. X. Qi, H. B. Huang, L.Chen, H. J. Wang, and C. K. Shen. Fast synchronization in neuronal networks. *Europhysics Letters*, 82(3), May.

[60] M. Rhouma and H. Frigui. Self-organization of pulse-coupled oscillators with application to clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):180–195, Feb 2001.

[61] K. Romer. Time synchronization in ad hoc networks. In *Proc. of the 2nd ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MOBIHOC 01)*, pages 173–182, Long Beach, CA, USA, Oct 2001. ACM Press.

[62] P. Rowcliffe, J. Feng, and H. Buxton. Clustering within integrate-and-fire neurons for image segmentation. In *Proceedings of International Conference of Artificial Neural Networks (ICANN 2002)*, volume 2415, pages 69–74, Madrid, Spain, Aug 2002. LNCS Springer-Verlag Berlin Heidelberg.

[63] A. Rowe, D. Goel, and R. Rajkumar. Firefly mosaic: A vision-enabled wireless sensor networking system. In *28th IEEE International Real-Time Systems Symposium, 2007. RTSS 2007*, pages 459–468, Tucson, AZ, Dec 2007.

[64] J. Sallai, B. Kusy, A. Prabal, and P. Dutta. On the scalability of routing-integrated time synchronization. In *Proceedings of 3rd European Workshop on Wireless Sensor Networks (EWSN)*, pages 115–131, Zurich Switzerland, Feb 2006.

[65] K. Sekiyama, Y. Kubo, S. Fukunaga, and M. Date. Self-organizing communication timing control for sensor network. *Complexity International*, 12, 2005.

[66] K. Sekiyama, K. Suzuki, S. Fukunaga, and M. Date. Autonomous synchronization scheme access control for sensor network. In *Proceedings of 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems,(KES 2005)*, volume 4, pages 487–495, Melbourne, Australia, Sep 2005.

[67] M. L. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC 2003)*, pages 1266–1273, New Orleans,Louisiana, USA, Mar 2003.

[68] S. H. Strogatz. Exploring complex networks. *Nature*, 410(6):268–276, Mar 2001.

[69] H. O. Tan, I. Korpeoglu, and I. Stojmenovic. A distributed and dynamic data gathering protocol for sensor networks. In *Proceedings of the 21st International Conference on Advanced Networking and Applications (AINA'07)*, pages 220–227, Niagara Falls, Canada, May 2007.

[70] J. Tate and I. Bate. An improved lightweight synchronisation primitive for sensornets. In *Proceedings of 6th IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 448–457, Macau, Oct 2009. IEEE Computer Society, Los Alamitos, CA.

[71] M. Timme, T. Geisel, and F. Wolf. Speed of synchronization in complex networks of neural oscillators Analytic results based on Random Matrix Theory. *Chaos*, 16:015108–015108–15, 2006.

[72] M. Timme, F. Wolf, and T. Geisel. Prevalence of unstable attractors in networks of pulse-coupled oscillators. *Phys. Rev. Lett.*, 89(15):154105, Sep 2002.

[73] M. Timme, F. Wolf, and T. Geisel. Unstable attractors induce perpetual synchronization and desynchronization. *Chaos*, 13:377–387, March 2003. Provided by the Smithsonian/NASA Astrophysics Data System.

[74] A. Tyrrell, G. Auer, and C. Bettstetter. Fireflies as role models for synchronization in ad hoc networks. In *Proceedings of 1st Bio-Inspired Models of Network, Information, and Computing Systems 2006, (BIONETICS)*, pages 1–7, Madonna di Campiglio, Dec 2006.

[75] A. Tyrrell, G. Auer, and C. Bettstetter. A synchronization metric for meshed networks of pulse-coupled oscillators. In *Proceedings of 3rd Bio-Inspired Models of Network, Information, and Computing Systems 2008, (BIONETICS)*, Hyogo, Japan, Nov 2008.

[76] C. van Vreeswijk and L. F. Abbott. Self-sustained firing in populations of integrate-and-fire neurons. *SIAM J. Appl. Math.*, 53(1):253–264, 1993.

[77] C. van Vreeswijk and L. F. Abbott. When inhibition not excitation synchronizes neural firing. *Journal of Computational Neuroscience*, 1(4):313–321, Dec 1994.

[78] P. Verma, A. K. Ghosh, and A. Venugopalan. Free-space optics based wireless sensor network design. In *Ist Symposium on Military Communications*, Prague, Czech Republic, April 2008.

[79] N. Wakamiya and M. Murata. Scalable and robust scheme for data fusion in sensor networks. In *the First International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-ADIT 2004)*, pages 412–427, Lausanne, Switzerland, Jan 2004. LNCS3141.

[80] D. L. Wang and D. Terman. Locally excitatory globally inhibitory oscillator networks. *IEEE Transactions on Neural Networks*, 6(1):283–286, 1995.

[81] D. L. Wang and D. Terman. Image segmentation base on oscillatory correlation. *Neural Computation*, 9:805–836, 1997.

[82] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, Jun 1998.

[83] G. Werner Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. Firefly inspired sensor network synchronicity with realistic radio effects. In *Proceedings of 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys'05)*, pages 142–153, San Diego, California, USA., Nov 2005. ACM Press.

[84] A. Winfree. Biological rhythms and the behavior of populations of coupled oscillators. *Journal of Theoret. Biol.*, 16:15–42, 1967.

[85] A. Winfree. *The geometry of Biological Time*. Springer-Verlag, New York, 1980.

[86] A. Winfree. *The Time of Biological Clocks*. Scientific American Press, New York, 1987.

[87] O. Younis and S. Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, Oct 2004.