ANALYSIS ON BEHAVIORS OF BOTNETS AND COMPLEX ROBOTIC NETWORKS

NUR ROHMAN ROSYID

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF ENGINEERING IN ELECTRICAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2012

KMITL-2012-EN-D-018-004

หัวข้อวิทยานิพนธ์	การวิเคราะห์พฤติกรรมของบอตเนตและระบบเครือข่ายเชิงซ้อนของหุ่นยนต์
นักศึกษา	นายนู โรมาน รอสยิด
รหัสนักศึกษา	51600151
ปริญญา	วิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2555
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร. ปิติเขต สู้รักษา

บทคัดย่อ

การศึกษาระบบเครือข่ายเชิงซ้อนเป็นปัญหาที่น่าสนใจในรอบสิบกว่าปีที่ผ่านมานี้ ทว่าการศึกษาดังกล่าวนั้นยังไม่ครอบคลุมกรณีของเครือข่ายหุ่นยนต์เชิงซ้อน โดยเฉพาะอย่างยิ่งผลการทดลองจริงยังไม่เคยมีผู้ใดทำมาก่อน งานวิจัยนี้เป็นการศึกษาพฤติกรรมระบบเครือข่ายเชิงซ้อนของหุ่นยนต์ทั้งในโลกเสมือนจริงและโลกจริง โดยในโลกเสมือนจริงจะศึกษาพฤติกรรมของเครือข่ายบอตเนต (Botnet) และศึกษาพฤติกรรมระบบเครือข่ายหุ่นยนต์ในโลกความเป็นจริง

ผลการศึกษาพบว่าบอตเนตในโลกเสมือนจริงมีพฤติกรรมการโจมตีเป็นแบบมีแบบแผนหรือเป็นแบบรายคาบ ซึ่งมีประโยชน์ต่อการตั้งรับการโจมตีระบบบนเครือข่ายอินเตอร์เนต ในขณะที่การศึกษาระบบเครือข่ายเชิงซ้อนของหุ่นยนต์ในโลกจริงมีพฤติกรรมหลากหลาย ตั้งแต่แบบคงค่า ทำนายได้ แบบรายคาบ และแบบอลวน (chaos) ทั้งนี้ ขึ้นอยู่กับความแรงของอันตรกริยาระหว่างหุ่นยนต์ สรุปได้ว่าพฤติกรรมของระบบเครือข่ายเชิงซ้อนของหุ่นยนต์เป็นแบบเฉพาะเจาะจง (deterministic) แม้ว่าจะอยู่ในภาวะอลวนซึ่งดูเหมือนจะเป็นแบบสุ่มสำหรับผู้สังเกตก็ตาม

Thesis Title	**Analysis on Behaviors of Botnets and Complex Robotic Networks**
Student	Mr. Nur Rohman Rosyid
Student ID.	51600151
Degree	Doctor of Engineering
Program	Electrical Engineering
Year	2012
Thesis Advisor	Assoc. Prof. Dr. Pitikhate Sooraksa

Abstract

Characteristics and properties of complex networks have been studied intently over the past decade. But, the study of behaviors of complex network in robotic field is very limited, especially in the real world experiments. This research deals with behaviors of mobile robots under the theme of complex networks both in real world and cyber world. The behaviors of the robots under investigation are the botnet (short for roBOT NETwork) in the cyberspace and the robotic network in the real space.

The study found that several behaviors are useful for alerting Internet users against botnet attack threats. To study behaviors of the mobile robots, a new model of the non-embedded complex robotic network is proposed. Simulation and experiment are also performed. In summary, this research confirms that behaviors of two kinds of robots forming complex networks as the dynamical system in the cyberspace and the real space, which are represented by the botnet and complex robotic networks, respectively, fall into one of characteristics of behaviors: fixed point, periodic, or chaos. The phenomena lead to the conclusion that the complex systems behave deterministically even though they seem to be appeared randomly to the observers.

ACKNOWLEDGEMENTS

And so those who were given knowledge may know that it is the truth from your Lord and [therefore] believe in it, and their hearts humbly submit to it. And indeed is Alloh the Guide of those who have believed to a straight path.

--the English translation of ***Al-Qur'an*** *[22:54]—*

The author would like to sincerely express his gratitude to his advisor, Dr. Pitikhate Sooraksa, for supporting his research and continuing encouragement along the entire period of his study. Guidance from his advisor either in his academic life and social life in Thailand are countless and unforgettable. The author would also like to thank to his Co-Advisor, Prof. Hiroaki Kikuchi, who gave many experiences in academic research during his doctoral sandwich program in Tokai University Japan.

The author would like to thank to all his committee members: Dr. Surapan Airphaiboon, Dr. Pitak Thumwarin, Dr. Veerapol Monyangkul, and Dr. Attasit Lasakul for the time spent reviewing this dissertation's document. Thanks also addressed to all laboratories' members who made the atmosphere more colorful. Special thank to Meow (Dr. Anurak Jansri) for unconditionally help and friendship during his study. The friendship and discussions of Pi Tor, Pi Yos, Jeng, Diew and Ajarn Doc (Dr. Kitdakorn Klomkarn) made his life in Thailand more enjoyable. The author would also be grateful to AUN/SEED-Net JICA for fully supporting his study under the scholarship program granted to him.

Finally, the author recognizes the powerful support from his lovely family, his lovely wife and lovely children which makes this achievement possible. Only Alloh S.W.T. (the one and only almighty God) that can give reward of theirs generosity and patience during the author study abroad. The author would like to thank to A. Etti Hendrawati (his wife), Farhan N. and Hanifah N. (his son and daughter, respectively) for everything.

Nur Rohman Rosyid

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (continued)

Page

TABLE OF CONTENTS (continued)

Page

TABLE OF CONTENTS (continued)

Page

LIST OF FIGURES

Fig. Page

LIST OF FIGURES (Continued)

Fig. Page

LIST OF FIGURES (Continued)

Fig. page

LIST OF FIGURES (Continued)

Fig. page

LIST OF TABLES

Table Page

Chapter 1

Introduction

1.1 Background

This dissertation is concerned with behaviors of mobile robots under the theme of complex networks both in real world and cyber world. Real world networks such as transportation networks, electronic circuit networks, ecological networks, and so on can be found everywhere all around us. And the cyber networks such as artificial neural networks, social networks, and so on are now spreading widely using the Internet infrastructure. The complex network phenomena [1-5] are inspired the development of this study in order to introduce the theme to field of robotics.

The understanding of the behavior of complex networks gives benefits for us. The Internet as a complex network not only brings advantages but also dangerous threats. The investigation and analysis on behaviors of the botnets—robot networks used by hacker for malicious purposes— recommend us to counter against their threats. On the other side of real space, the investigation and analysis of the behavior of complex robotic networks can give innovations for the industrial applications. Remark that the term "botnet" is short for roBOT NETwork and "robotic network" is for real space. Would it be a connection via the complex network for robots as the terms are coincident in both for cyberspace and real space?

To compare different and common features of botnet traveling through the computer network and robots traveling around the robotic network, we first consider the physical structures of both types of robots. In cyberspace, the terms of a robot or a robot-network is well known as a *botnet*. The botnet is a set of infected and compromised computers (bots/robots) connected to the Internet which is controlled remotely by an authorized user (botmaster), and employed for nefarious purposes [1-5]. Whereas in real space, the terms of robot refers to electro-mechanic device

that performs tasks autonomously or semi-autonomously. Especially in this research, a two-wheeled differential-drive mobile robot is used. This mobile robot is popularly utilized in the research field of mobile robotics due to its simplicity. It performs a simple move such as forward, backward, turn right and turn left, and be able to communicate primitively. Convocation of the mobile robots forms a robotic network. This type of scheme is the theme represents a complex network in real space via interaction among them.

Robot networks in both of cyberspace and real space have different features. Botnets propagate through computer networks connected by wired or wireless links. Botnet, physically and behaviorally, are not tangible objects that can be seen directly with human eyes. Its behaviors can be seen indirectly from chaos or damages as a result. Meanwhile the robotic networks in real space consist of mobile robots are connected commonly used wireless links in particular way. And of the robotic network can be physically and behaviorally seen directly with human eyes. The topics are discussed more below.

1.1.1 Botnet

Internet malware threats have increased and have become extremely sophisticated, they are also becoming commercialized [6, 7]. Highly organized and coordinated attacks by botnets are able to make malicious activities such as distributed denial of service (DDoS), e-mail spam, and click fraud [7]. To scale up the botnets, bots infect additional computers by sending malware, using various strategies such as self-replicating worms, email viruses, or password guessing. The botmaster can then send arbitrary commands to the botnet to take control of victims. These commands are issued using one of two control mechanisms [2, 6]. The first method involves a centralized architecture, in which the bots in the botnet communicate with a central Command-and-Control (C&C) server. Most botnets use Internet Relay Chat (IRC) servers as C&C servers. The C&C server forwards commands

received from the botmaster to all bots in the botnet. In the second method, botnets use a distributed control mechanism via a Peer-to-Peer architecture.

Fortunately, botnet activity can be traced by the observation of malware footprints of several bots spread across the network. This method is used in the ``honeypot" system. In [4], McCarty gave an example of the implementation of a honeypot that captures useful data about computer attacks in the network. The honeypot is a decoy host pretending to be a vulnerable computer that looks attractive to the attackers, i.e., a host dedicated to receiving attacks [5]. The honeypot records every inbound packet as an item in an access log, comprising Timestamps, Honeypot ID, Source/Destination port number, Source IP address, Hash value (SHA1), Malware name[i], and Malware file name.

In [8], Wilson reported malware developers improve their technique using a modern skill, open-source, collaborate modular and reusable malicious codes for designing newer versions of malware for different criminal purposes. Overcoming the highly organized and coordinated malware threats by botnets on the Internet is becoming increasingly difficult.

For the foregoing threats, we have been motivated for contributing to overcome the problem. A honeypot is a powerful tool for observing and catching malware and virulent activity in Internet traffic. In this research, we investigated 94 independent honeypots that have observed malware traffic on the Japanese tier-1 backbone. The observations were coordinated by the Cyber Clean Center (CCC). The CCC DATAsets for 2009 comprise the access logs of attacks between May 1, 2008 and April 30, 2009. Because botnets use systematic attack methods, the sequential behavior of malware downloaded by honeypots have particular forms of coordinated pattern. One problem is the difficulty in identifying particular patterns from full yearlong logs because the dataset is too large for individual investigations. Thus, we

[i] The malware name is derived from the malware signature used by commercial anti-virus software (Trend Micro).

propose the use of a data-mining algorithm to overcome this problem. We implement the *PrefixSpan* algorithm to analyze botnet-attack logs. This is an underline of the scope of study for botnets under investigation.

1.1.2 Complex Robotic Network

Behaviors of complex networks in the real space have a wide range of possibilities stemmed from steady-state fixed point, periodically and sequentially repeated pattern, and continuous route to chaos. When a small difference in the initial condition is applied to a deterministic system, then a nearby trajectory diverge exponentially fast. Thus, this kind of system is called a chaotic system. Chaotic systems have attracted attention from researchers due to potentially real-world applications such as encryption technique [9-11], secure communication [12, 13] and in robotics. Exploration of the chaotic dynamics in robotics ranging from laboratory to commercial scales has been achieved [14]. Chaos system is also involved in the design path-planning as algorithms for stationary obstacle avoidance with chaotic dynamics using a neuron model [15]. In general, designers use a standard map to generate a trajectory of mobile robot [16]. However, non-repeatability trajectory of chaotic dynamics guarantees a mobile robot to navigate throughout the whole workspace [17]. Hence, implementation of chaotic dynamics for a robot searching and patrolling is alternatively suitable [18, 19]. For better areas coverage, multiple robots working together can cover more workspace faster than that of using the single one [17, 20]. To implement such chaotic behaviors, an ordinary mobile robot is embedded with a chaotic equation similar to the initial work of Nakamura and Sekiguchi [21]. Consequently, the robot moves chaotically following the dynamics of the equation.

Cooperation of chaotic mobile robots in the network becomes important due to robustness in term of a robotic network and effectiveness in term of the area coverage planning. Some researchers propose methods based on chaos

synchronization [17, 20]. However, this method has a short coming that the duty cycles of the robots' controllers in the real application are intensively burdened by the computational cost of running chaotic equation while performing chaos synchronization at the same time. A challenge appears to answer an emerged question is how to make a network of cooperation among ordinary mobile robots behaved chaotically without chaotic equations or extra chaotic circuits embedded into the robots. In doing so, a network must become a special complex network in which the dynamics of the global behaviors of the system emerged from a strongly interconnected among the agents. In fact, networks of dynamical systems have been used to model a wide range of complex systems [22]. Dynamics behavior such as chaos not only can be found in large networks, but also in a minimum complex network composed of the number of elements as small as four indicated by a wide variety of chaotic strange attractors [23]. This body of knowledge enables us to transform a group of four ordinary robots into a minimum complex network possessing chaotic behavior.

With the aforementioned motivation, a part of this research aims to bridge the gap between the theory of complex network and the implementation of chaotic mobile robots. Computer simulation and experiments have carried out to study the chaotic behavior of complex robotic network that established with strong interactions among robots obey the inverse square law of distance in the real space. This summarizes the scope of work under investigation.

1.2 Objectives

The objectives of this research are:

(1) to discover the sequential patterns or behaviors of malware attack in botnets;

(2) to discover a new method to alert the network users against the threats of the spread and the malaware attack from botnet, and

(3) to develop and study chaotic behaviors of a non-embedded mobile robotic network.

1.3 Contributions

Our contributions can be divided into two domains. The behaviors of the robot in the cyberspace and the real space have been investigated. This research brings contributions for each domain of botnets and complex robotic networks.

1.3.1 Contributions in Botnet

1) Detect the coordinated malware servers that are source of frequent sequential attack patterns.
2) Classify and characterize the frequent sequential attack patterns based on IP-address and timestamps.
3) Classify the group of sequential attack patterns based on the time duration of attacks.
4) Classify the sequential attack patterns based on how some common patterns are in attacks over computer networks.
5) Recommend some potential applications such as Intrusion Detection and Prevention System (IDPS), botnet firewall, and tracking botnets.

1.3.2 Contributions in Complex robotic network

1) Design a mathematical model of the non-embedded complex robotic network.
2) Design and construct a physical two-wheeled non-embedded complex robotic network.
3) Carry out the simulation of the non-embedded complex robotic network.

4) Compare the embedded and non-embedded chaotic mobile robot in terms of its area coverage.
5) Perform experiment on the non-embedded complex robotic networks for confirming the correctness of the proposed model.

1.4 Dissertation Outline

As mentioned earlier, this research is concerned with behaviors of vulnerable robots in complex networks both in cyber and real space. The study carries out simulations, experiment, and analysis on botnets and complex robotic networks. In this chapter, we have introduced the background of the robots in both domains. Objectives and contributions have also been described.

Chapter 2 presents literature review. Several methods for exploring behaviors of malware attack in botnet and various approaches for detecting and analyzing botnet attacks are introduced. Chaos system and applications, particularly in the embedded chaotic mobile robots are also discussed.

Chapter 3 discusses the analysis on behaviors of sequential patterns of botnet attacks. Malware traffic on the Japanese tier-1 backbone observed with 94 independent honeypots is investigated. Data mining algorithm namely *PrefixSpan* is implemented to reveal the behavior of botnet attacks. The sequential attack patterns of malware in botnet are exposed. Classifications of malware attacks are also described.

Chapter 4 proposes a model of the complex robotic network and the simulation. We explore behaviors of the complex robotic network in several possible different conditions and scenarios. Simulation and numerical analysis of the dynamics of the proposed model are presented. Chaotic behaviors are proved with the existence of the positive value of the largest Lyapunov exponent. A possible

potential application of complex robotic network for area covering is also performed and evaluated.

Chapter 5 presents the realization of experiment of the non-embedded complex robotic network. The experimental setup and the results of the non-embedded complex robotic network are shown and discussed. Hardware, software, and other components of the robot are illustrated. Estimations of the largest Lyapunov exponents from time series data, which is generated from the robots' trajectories, are presented at the end of the chapter.

Chapter 6 gives conclusions of this research. It summarizes the contributions and recommends some promising future works.

Chapter 2

Literature Review

2.1 Introduction

This chapter deals with two main parts. Firstly, we discuss about botnets and several methods for exploring behaviors of malware attack in botnet. Secondly, we mention about chaos system and applications on chaotic mobile robots. The former part of this chapter is limited and focused on various approaches for detecting and analyzing botnet attacks from a traffic flows or dataset. The honeypot technology for capturing malware activity is also discussed. The latter part deals with the chaotic robots traveling in real space.

In the real space, phenomena of chaos emerge from a deterministic system that exhibit non-periodic behavior that depends sensitively on the initial conditions. Although the interesting applications of chaotic systems in the complex world are diversified, this chapter focuses merely on the applications of chaotic field of chaotic mobile robots. We will find out later on the next chapters that we can substitute the similar behaviors of this system by using a group of ordinary robot forming a complex system.

2.2 Honeypot and Various Approaches of the Analysis on Botnet Attacks

To expatiate upon the theme, we first begin with botnet in this subsection and then review the chaotic behaviors of mobile robots to form a complex network in the next section. The term *bots* is sort of robots, in the cyberspace it is a self-regulating malicious code that functions as a slave for botmasters [5, 24]. A collection of compromised computer (bots) in the Internet that are controlled by botmaster for a bad purpose is called botnet [6]. Usually bots spread through the Internet via email

in mail groups or mailing lists. When the user executes the malicious program by clicking an attached file at the email, it will automatically infect the vulnerable computer. Another method is network users click a link in a web page that the web server has defaced by hackers [3, 25]. A total bandwidth and a huge size of botnet become a terrible threat and it hard to against. In [4], McCarty mentioned that IRC server reported 4.752 hosts were joined in the channel. In addition, 15.164 unique hosts occupied the network traffic over a period about 10 days. Suppose that bandwidth for each bot is 56kbps with botnet scale around 15,164 bots attacking at the same time would give result in 850 Mbps at its victim. It is effective way to cripple the target such as any e-commerce site.

2.2.1 Honeypot

To fight against the botnet, we must have useful information about the botnet attack. There is a powerful tool that is called a *honeypot*. This is a kind of a host that pretends to be a vulnerable computer for receiving attacks. This trick addresses for capturing malware activities in a machine that has been compromised by attackers. Datasets have been captured by honeypots are valuable information about techniques, strategies and motivations of attackers [5, 26, 27].

Vulnerabilities give a crack to the attacker to infiltrate into a victim's computer after the attacker found its weaknesses. Furthermore, some computer worm such as *Warhol worm* is able to attacking all most vulnerable victims under an hour, probably no more than 15 minutes which is impossible for making counter responses immediately [28]. To learn how the attacker utilizes vulnerabilities a computer system one can develops a host in the network to observe what happen in it. If the host system is dedicated without any purposes, then anytime the outsiders want to try for contacting it, they seem to be suspected. The important values come up from the host being attacked, probed, compromised; and thus the host such this is called a honeypot [2]. Defensive approach most have implemented

in the network security that aims to keep the attacker out of the system. Unlike the defensive approach, honeypot is about keeping the attacker inside of the system, and behaves that seems to be attractive to the attacker [5].

There are two categories of honeypots based on the interaction between honeypots and the attacker. A high-interaction honeypot provides a real system that the attacker can interact with. In contrast, a low-interaction honeypot simulates only some parts of network services. Another difference is physical and virtual honeypots. A physical honeypot is a real machine on the network with its own IP address. A virtual honeypot is simulated by another machine that responds to the network traffic sent to the virtual honeypot.

A honeypot has several advantages such as data value, resources, and simplicity. Data value arises from the honeypot because any connection inbound to the honeypot is most likely a probe or attack, and honeypot only harvest small amounts of data, but high-value data.

Resources are another issue in network monitoring system. Resource exhaustion can happen when a security resource has not continually works because its resources are overwhelmed. Because the honeypot captures and monitors a few events, the honeypot typically do not have problems of the resource exhaustion. The honeypot requires minimalist resources to be developed, low investment of money in the hardware. It does not require the new technology, vast amounts of RAM or chip speed, or large disk drives.

The simplicity is the best achievement to be considered as the biggest advantage in the honeypot. The honeypot does not need a sophisticated algorithms to develop, and signature database to maintain. One can drop the honeypot somewhere in his/her organization and then sit back and wait. If outsider connects to the honeypot, check it out.

Several disadvantages also come to the honeypot such as narrow field of view, fingerprinting, and risk. It does not replace any security mechanisms, in other words it is a complement for enhance the overall security architecture. The honeypot only monitors a particular activity which is predefined. A honeypot has a microscopic effect on the value of the data harvested. It only focuses closely on the data of known value.

The honeypot probably cannot behave exactly the same as the real system. Sometime the attacker is able to catch an anomaly behavior of the honeypot. For example, the honeypot may be designed to emulate an NT IIS Web server, but the honeypot also has certain characteristic that is identified as a Unix Solaris server.

The honeypot can introduce a risk of its environment. When the honeypot has compromised, it can be used to attack, infiltrate, or harm other systems or organizations. Each honeypot has different level of risk. The simpler the honeypot, the less the risk; it depends on how the honeypot was built and deployed.

2.2.2 Some Approaches of the Analysis on Botnet Attacks

The honeypot gives benefits to those who work for network security. It can distract attackers from more valuable machine on the network; provides an early warning about a new attack and exploitation trends; or allows in-depth examination for attackers during and after the exploitation against the honeypot.

The study of datasets collected from honeypots provides valuable knowledge on the attacking behavior of botnets. Heuristic techniques for the detection of malware involved in botnet coordinated attacks provide useful information for determining the characteristics of and relationships between botnet attacks. In [29], Kuwabara, et al. reported that the botnet sends packets containing particular messages indicating malware. A message such as *MZ, PE, DOS,* and *"!this program cannot be run in DOS"* or *"!Windows Program"* appear in files downloaded

use Trivial Files Transfer Protocol (TFTP). A compromised computer establishes communication to the C&C server in order to join in the botnet. A message such as *NICK* or *JOIN* was captured on the traffic flows when the IRC protocol is used. These messages indicate that the bot tried to apply its nickname and want to join into the channel. Along his investigation 13 kinds of malware have exposed. Generally bot attacks pass three phases [3]: (1) *Scanning/Searching*, scanning for vulnerable computers, (2) *Infection*, bots spread malware for infecting victims (indicated as vulnerable computers), successfully infect victims has an effect of victims become new bots, (3) *join to the channel*, new bots submit themselves to the botmaster and ready for receiving commands.

Nowadays the botnet developer continuously improves techniques more sneaky. Most popular centralized technique of botnet used IRC channels [30], then expanded to the popular applications such as HTTP protocol [31]. But the centralized technique has a weakness of single point of failure. Considering the weakness, a robust botnet implements a distributed P2P-architecture instead of the centralized one, if the one of the C&C server peer collapses, then others can substitute it [6, 25].

Many threats come from the Internet such as Deniel-of-Service (DDoS), click fraud, and phishing are parts of botnet attacks. These distributed attacks have triggered by botmaster through the Internet. Guofei Gu, et al. [32] explained a mechanism for monitoring network perimeter to detect the Internet malware infections that caused by botnet. Because botnet uses mechanism C&C channel to communicate and distribute attacks, it has two-way communication or dialog among botmasters and bots. During the infection process, the communication sequence intently occurs. The system monitor is called *BotHunter*. He proposes a method that called dialog correlation strategy, the process of bot infections are modeled as a group of traffic flows such as target searching, infection exploit, binary egg download and execution, command and control channel establish, and outbound scanning that

bartered between internal host and external entities. If there is a correlation among inbound scan, intrusion alarms, and outbound communication pattern, then it's a strong indication that a local host is successfully infected.

In [33], Guofei Gu, et al. presented a general framework of botnet detection that not subject to botnet C&C protocol and construction, and without any prior information of botnets. The framework uses clustering and cross-correlation techniques to identify a botnet. A filtering process is needed to eliminate unnecessary traffic flows. The next step is to cluster the flows based on similarities between malicious and communication activities. If a host belongs to both clusters, then it is strongly identified as part of a botnet. Another approaches based on clustering, Thonnard and Dacier in [34] utilize graph-based clustering method for investigating datasets of Internet network traffic collected by honeypots based on similarities of time-signature. Closely similar patterns of time-signatures clustered within certain time duration at several different sensors reveal coordinated activities come from the same root cause. They noted that the "shape" of time series may have significant information such as a routine diurnal/nocturnal activity pattern probably a sign of botnet propagation. They have found that an appropriate similarity measure on time series analysis enables the identification of several worms and botnets activities.

Another attempts to detect botnets based on network traffic flows was study by Lu Wei et al. [35]. Lu Wei et al. reported noted about 40% unknown network flows were found on a large-scale WiFi-ISP network over a half-year period that might be caused by malicious activities or a new botnet traffic. To identify and classify the unknown flows, they proposed an automatic probing of botnet communities. Using cross-association clustering and *N-gram* algorithm for investigating the normal network traffic flows on large-scale WiFi-ISP network. Its process needs three steps to accomplish the goal. First, classify the input network flows based on a payload signature to get the *unknown* flows. Secondly, examines the *unknown* flows based

on cross-association clustering method to classify it into application communities such as P2P community, HTTP Web community, Chat community, Data Transfer community, Online Games community, Mail Communication community, Multimedia (streaming and VoIP) community and Remote Access community. Finally, using *N-Gram* algorithm to define whether the network flows are generated by humans or bots. They claim successfully achieve an automatic application for classifying network application communities and a generic method to distinguish malicious activity between humans and botnets.

Because botnet is controlled by botmaster in a systematic and coordinated way, botnet generates activities with similar behavior patterns. Let us take a look at the spam activities. We are annoyed with receiving email spam for advertising some products market. A lot of spam emails are routinely sent to our inbox every day. It is quite possible that spammers systematically send the spam emails to targets. Nowadays spammers have choices for spamming, because botnets are commercialized and powerful for spreading spams [7, 8, 24]. Husna et al. in [36] investigate the behavior patterns of spammers based on the core similarities within spamming, particularly their temporal characteristic. Principal component analysis is applied to a feature set to determine which characteristics are important for the highest diversity in the spamming patterns, such as the active time, the content length, and frequency of emails. Clustering methods are then used to classify spammers into botnet groups based on behavior similarities. They claim that this method enables us to recognize botnets precisely from their similar behavior when spamming a domain.

2.3 Chaotic Mobile Robots

A chaotic mobile robot is a mobile robot that behaves chaotically based on a chaotic system embedded into the mobile robot controller [14, 17-21, 37-40]. The chaotic mobile robot usually is implemented into two-wheeled mobile robot.

Because its simplicity and ease of controlling, the two-mobile robot becomes popular in chaotic robotic field. Fundamental of chaos and mobile robots will be given as follows.

2.3.1 Chaotic System

Chaos is phenomenon that deals with inherent unpredictability in the behavior of complex natural system [41]. It is a part of foremost subject known as a dynamics. This subject relates to dynamically changes of the system [42]. Behaviors of the chaotic dynamic system externally seem to be non-periodic and random. However, this phenomenon comes up from deterministic systems generated by physical or mathematical law, instead of noisy driving forces. Chaos is famous with its definition as a deterministic system exhibits non-periodic behavior that depends sensitively on the initial conditions.

2.3.1.1 The Dynamic System as an Ordinary Differential Equation (ODE)

The dynamical system can be expressed mathematically as a differential equation. This subsection deals with a temporal behavior represented by ordinary differential equations (ODEs). General framework for ODE is stated below [42]:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, \cdots, x_n) \\ &\vdots \\ \dot{x}_n &= f_n(x_1, \cdots, x_n) \end{aligned}, \tag{2.1}$$

the over dots denote differentiation with respect to t. Thus $\dot{x}_i \equiv dx_i/dt$; it says that how the system evolves in time. And then the problem at hand is described as functions $f_1, \cdots, f_n$. The number of variables involved in the system is denoted with n and the variables are stated as $x_1, \cdots, x_n$.

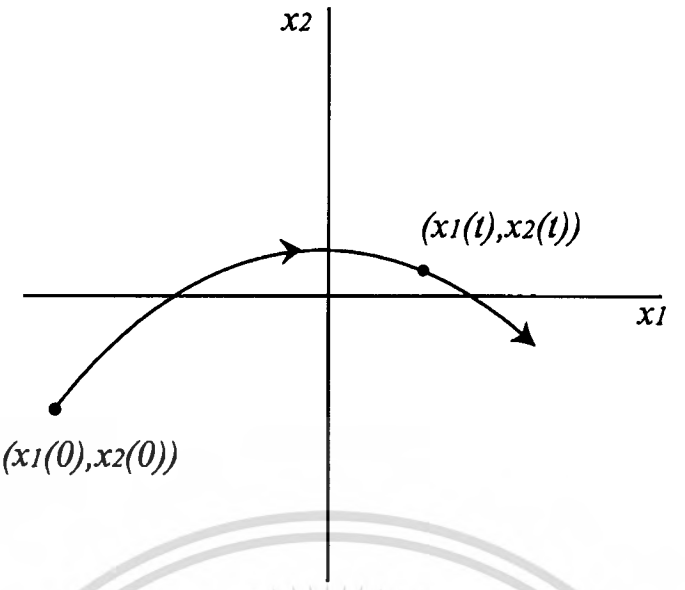


Fig. 2.1 Trajectory on phase space x_1-$x_{(1+k)}$ where k=1

In a nonlinear system, the nonlinearity can be linearized by a small angle approximation $f(x) \approx x \; for \; x \ll 1$. This assumption converts the problem to a linear one. The solution would be a pair of functions $x_1(t)$ and $x_{(1+k)}(t)$, where $0 < k < n$. Figure 2.1 shows geometrical representation of the solution that can be done by plotting the solution $(x_1(t), x_{(1+k)}(t))$ corresponding to a point evolved along a curve on the abstract space with the coordinate$(x_1, x_{(1+k)})$.

2.3.1.2 Equilibrium and Stability

The term *equilibrium* of a system is a state of the system which does not change [43]. To estimate the equilibrium of the system in the dynamical system, it can be done by setting all derivatives to zero. *Fixed points* are denoted by x^*, defined by $f(x)=0$. The synonym of fixed point is steady, constant or rest of solutions. If $x=x^*$ initially, then $x(t)=x^*$ for all time.

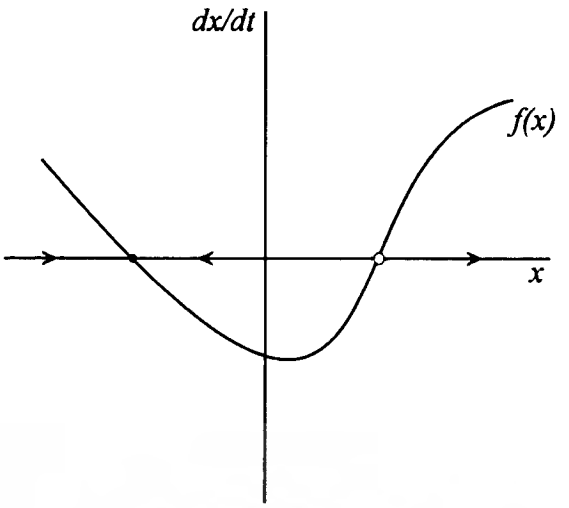


Fig. 2.2 Vector field on phase portrait of *f(x)*

Figure 2.2 shows a vector field of a one-dimensional system $\dot{x} = f(x)$ on the real line, it simply draws the graph of *f(x)* and then use it to sketch the vector field, where the xy-axis are x and $\dot{x}$, respectively. The solution of $\dot{x} = f(x)$ is started from arbitrary initial condition x_0 and is observed how it is carried along the flow. A path of the solution is called trajectory based on x_0, and the diagram as shown in Fig. 2.2 is called a *phase portrait.*

If $\dot{x} > 0$, then the arrows flow to the right, otherwise to the left. The other state is $\dot{x} = 0$ which is no flow, so that's why is called fixed point. As shown in Fig. 2.2, there are two kinds of fixed points; solid black dots represent stable fixed points (also often called attractors or sinks, because the flow is toward them) and open circles show unstable fixed points (known as repellers or sources). The equilibrium defines stability of a system, if all sufficiently small disturbances away from a point dump out in time, then it is called stable or equilibrium point. In contrast, if disturbances grow in time, then it is called unstable point.

2.3.1.3 Chaos Definition

Chaos is the phenomenon of occurrence bounded aperiodic long-term behavior in completely deterministic nonlinear dynamical systems with high sensitive dependence on initial conditions [44, 45].

Aperiodic long-term behavior means when $t \to \infty$ trajectories do not converge to equilibrium points, periodic points or quasiperiodic orbit. It forces an open set of initial conditions for leading to aperiodic trajectories; or probably, it given a random initial condition and thus the probability of aperiodic trajectories occurred is nonzero.

Deterministic means the system does not be influenced by any random or noisy inputs or parameters. The aperiodic behavior comes up from the nonlinearity of the system itself.

High sensitive dependence on initial conditions means as time goes on, trajectories nearby points diverge exponentially fast, i.e., the system has a positive Lyapunov exponent number.

2.3.1.4 Attractor and Strange Attractor Definition

The chaotic behavior of the system can be observed on phase space. When the chaotic behavior comes up on an attractor, it will be the most interest to be observed. Generally speaking, an attractor is all trajectories of a set near neighbor points converged to a chaotic region [44, 46]. An attractor can be defined as a set *A* with the following properties:

a) *A* is an invariant *set*: any trajectory *x(t)* that starts in *A* stays in *A* forever.

b) *A* attracts an open set of initial conditions: there is an open set *U* containing *A* such that if $x(0) \in U$, then the distance from *x(t)* to *A* tends to zero when $t \to \infty$. This condition confirms that *A* attracts all trajectories that start sufficiently close to it. The largest such *U* is called a basin of attraction of *A*.

c) *A* is minimal: *A* does not have a subset that confirms conditions ***a)*** and ***b)***.

A simple attractor can be explained as points and circle like curves called limit circles (an isolated closed trajectory). Attractors that exhibit sensitive dependence on initial conditions, and have a great detail and complexities, it is called a strange attractor. For example, Lorenz's attractor is the famous one of as a strange attractor, as shown in Fig 2.3.

Fig. 2.3 The chaotic strange attractor of Lorenz system

2.3.1.5 Quantifying Chaos

A chaotic system behaves sensitively dependent on initial conditions which mean two trajectories starting very close together will rapidly diverge from each other, and thus have totally different futures. Property such this qualitatively can be specified and characterized by existence of the largest positive Lyapunov exponent number. The Lyapunov exponent tells us how the system evolves in time and what happens to nearby trajectories as time goes on [47, 48]. One positive Lyapunov exponent in the system is enough to claim it is a chaotic system.

In [49], Sprott explained step-by-step for the calculation of the largest Lyapunov exponent from a dynamical differential equations. First, start with any initial point $(x_0, \cdots, x_{n-1})$, one may iterate until the system is converged into the attractor. Second, select nearby points $(y_0, \cdots, y_{n-1})$ with small separation $\mathcal{E}$ at time t. Third, define $\Delta x_t = (x_t - y_t, \cdots, x_{t+n-1} - y_{t+n-1})$ and let $|\Delta x_0| = \varepsilon$. Fourth, evaluate $\left|\frac{\Delta x_1}{\Delta x_0}\right|$ and record. Fifth, readjust the vector $\Delta x_1$to the length of $\mathcal{E}$, then the new neighbor $(y_1, \cdots, y_n) = (y_1, \cdots, y_n) + \Delta x_1$ is increment on time period along with $(y_1, \cdots, y_n)$. This process is repeated, and the largest Lyapunov exponent is estimated from the average of the logarithm of the ratios:

$$\lambda_{max} = t^{-1} \sum_{i=0}^{t-1} ln \frac{|\Delta x_{i+1}|}{|\Delta x_i|}, \quad (2.2)$$

where t is the number of iterations of the flows over which the average is taken.

To estimate the largest Lyapunov exponent from the experimental data, we cannot be directly as the calculation of dynamical differential equations. Due to the experimental data usually one dimensional time series, it has to be reconstructed into the attractor that properly unfolded. Time-delay coordinate of a phase space is frequently implemented to overcome this matter [50-54]. Given a scalar time series $x(t_0), x(t_1), x(t_2), \cdots, x(t_i)$ where $x(t_i) \in R$ is a measurement taken at time $t_i = t_0 + i\Delta t$. One dimensional time series can be embedded into a higher dimensional space m using time-delay embedding τ yields vectors $\hat{X} \in R^m$ as

$$\hat{X}_{(t)} = \left(x_{(t)}, x_{(t+\tau)}, \cdots, x_{(t+(m-1)\tau)}\right). \quad (2.3)$$

The time-delay embedding τ defines the quality of the reconstruction. When τ is chosen too small the trajectory of the time-delay coordinate appears to be a straight line with angle 45^0 on the phase space, and thus $x_{(t)}$ and $x_{(t+(m-1)\tau)}$ will be indistinguishable. In opposite, if τ is too large, then the trajectory will fill out all spaces in phase space which is not represent the true attractor. Fraser et al. [55] recommends for using the first minimum of the mutual information to choose a proper time-delay embedding.

Whereas a choosing of the minimum embedding dimension is another case, if it follows $m>D$ where D is a fractal dimension of the attractor, then the proper time-delay coordinate will carry out the same dimension as the original attractor [52, 56]. To estimate an appropriate embedding dimension m for a vector $\hat{X}_{(t)}$, Kennel et al. [53] proposes a method which is based on the number of false nearest neighbors points. The idea is the time-delay coordinates in R^m unfolded properly when the number of false nearest neighbors is falling down to zero. In [57], Cao implements the mean value of the number of nearest neighbor to estimate the minimum

embedding dimension m. The mean value stayed constant when the embedding dimension m_0 has increased to the higher dimension m_1, i.e, if m_0 increases to m_1 and the mean value of the number of nearest neighbor stays constant at $m_{(1+i)}$. And the term m_0+1 is the minimum embedding dimension.

From Eq. (2.3) with a proper m and τ as described, Wolf [58] explains how to estimate the largest Lyapunov exponent number. Initially denote a fiducial point and its nearest neighbor in phase space and calculate the Euclidean distance between two points $L(t_0)$. As the two points evolve at t_1 , the Euclidean distance becomes $L'(t_1)$. The evolution time is kept in a time short enough ensuring to probe the structure of attractor in a small area. The large time evolution will lead an error estimation of λ_{max}. Next, two criteria, a small distance $L(t_1)$ from evolved fiducial point, and a small angular distance θ_i between the evolution and replacement point are considered to pick up new data point. This process is iterated until the fiducial trajectory has scanned all data file. By doing so, the largest Lyapunov exponent can be estimated as

$$\lambda_{max} = \frac{1}{t_M - t_0} \sum_{k=1}^{M} \ln \frac{L'(t_k)}{L(t_{k-1})}, \tag{2.4}$$

where M represents the total number of replacement steps. Figure 2.4 shows the illustration of the evolution and replacement method used to estimate the largest Lyapunov exponent from experiment data

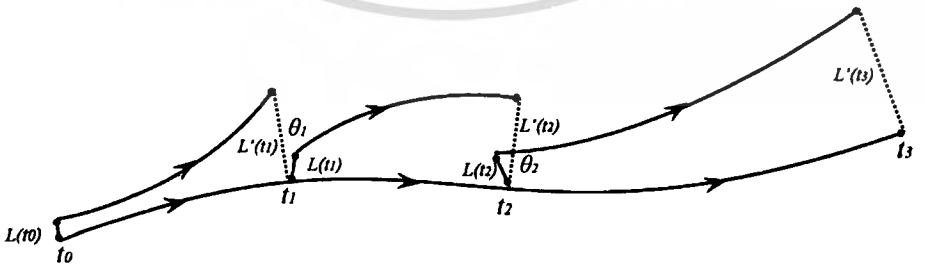


Fig. 2.4 Illustration of the evolution and replacement of Wolf's method to estimate the largest *Lyapunov* exponent number

2.3.2 Two-Wheeled Mobile Robot

A wheeled mobile robot is widely used from a high school workshop to a university labs scale. Discussion about wheeled mobile robots is widely available in many literatures. It has many types based on drive technology such as differentially drive, car-like, omnidirectional and syncho drives. This chapter focuses on the differentially drive or two-wheeled mobile robot.

2.3.2.1 Differentially Driven Two-Wheeled Mobile Robot

A differentially driven wheeled mobile robot (WMR) has two main parts that make it different to the other types. A WMR is built by two driving wheels, and one or more castor wheels in the front or rear side to support the balance of the robot's body. According to two wheels independently installed in the body structure of the robot, it requires a good coordination between two wheels of the right and left to achieve the desired motion. Benefits from the structure and configuration are its simplicity and able to adapt in many different conditions. Moreover, it is ease to construct and to control [59]. Figure 2.5 shows a general structure of the two-wheeled mobile robot.

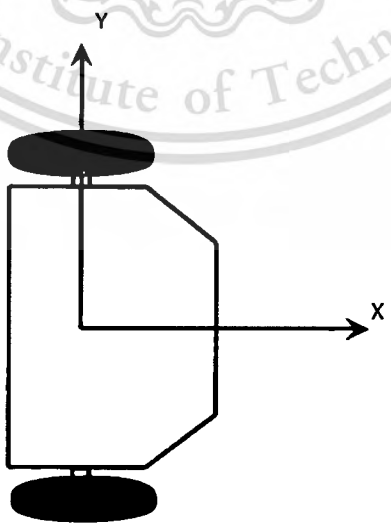


Fig. 2.5 The general structure of two-wheeled differential driven mobile robot

The concept of differential driven to control the motion of the robot arises from differential rate and rotaries of two wheels of robot on the right and the left. If both wheels roll at the same speed, then the robot will move forward in the straight line. The robot will move backward by rolling both wheels to the opposite direction. By slowing down or speeding up one wheel and keep the other in the constant velocity will make the robot turn right or turn left.

2.3.2.2 The Kinematic Model of the Two-WMR

Kinematics is the study of mathematics of the motion to deal with only geometry aspects and mechanical behaviors without any consideration of forces that affects the motion. This subject is important for designing and controlling the wheeled mobile robot. Two-WMR has constraints attached on its mobility, and this concept leads us to understand the mobile robot motion.

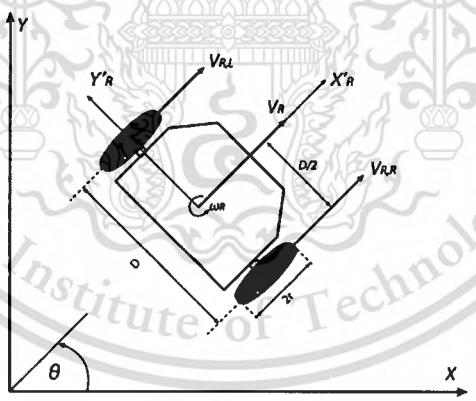


Fig. 2.6 Kinematics and local coordinate system where D is a diameter of the robot, ω_R defines an angular velocity, and $v_R, v_{R,R}, v_{R,L}$ are robot's velocity, right and left velocities, respectively

Figure 2.6 shows kinematics and the local coordinate system of the two-wheeled mobile robot. The robot moves with velocity v_R and angular velocity ω_R. The robot motion on the workspace is controlled from differential velocity between

the right wheel $v_{R,R}$ and the left wheel $v_{R,L}$. The conversion of the robot's velocity v_R to both velocities $v_{R,R}$ and $v_{R,L}$ are stated in Eqs. (2.5) and (2.6) below [60],

$$v_{R,R} = v_R + \frac{D.\omega_R}{2}, \; v_{R,L} = v_R - \frac{D.\omega_R}{2}, \tag{2.5}$$

Solve for *vR* and ω_R

$$v_R = \frac{v_{R,R}+v_{R,L}}{2}, \qquad \omega_R = \frac{v_{R,R}-v_{R,L}}{2}. \tag{2.6}$$

The curvature ρ of the trajectory is obtained from the holonomic relation between the velocity v_R and the angular velocity ω_R, then,

$$\rho = \frac{v_R}{\omega_R}. \tag{2.7}$$

The coordinate transformation is important part of the study of a rigid body motion on the global coordinate. The robot as a rigid body has a fixed coordinate system (*x'*,*y'*) and has the common origin (*0,0*) with the global coordinate system (*X,Y*) on workspace. Wherever the robot moves and rotates, its new coordinate is limited to the global coordinate system *X* and *Y*. The motion of the robot in a two-dimensional workspace must respect to the restrictive equation. Assume a point *A* in the *X-Y* plane has a coordinate *A*(*x,y*) as shown in Fig. 2.7. A new coordinate axes (*X'-Y'*) on the same workspace can be created by simply rotating the original coordinate axes with an angle θ at its common origin, that is (*0,0*). Suppose a point *A*(*x,y*) is transformed into a new coordinate system (*X'-Y'*) plane. Thus, the transformation becomes

$$x' = x\cos\theta + y\sin\theta$$
$$y' = -x\sin\theta + y\cos\theta \quad (2.8)$$

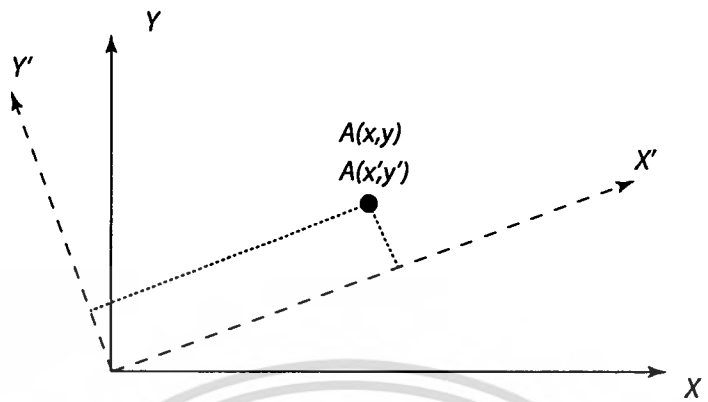


Fig. 2.7 Coordinate transformation, where *X-Y* plane is the global coordinate system and *X'-Y'* is the local coordinate system of the central mass/robot

Positions of the robot along its trajectory can be determined with an assumption, at time *0* the robot's position is $P_0=[x_0, y_0, \theta_0]^T$. If the robot moves with the velocity v_R and angular velocity ω_R, then the next position *P* is a function of $P=f(P_0, v_R, \omega_R, dt)$.

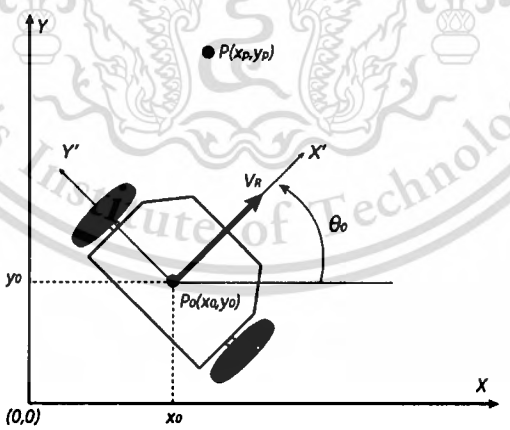


Fig. 2.8 The coordinate system of the moving robot, where $P_0(x_0, y_0)$ is the initial position of the robot, it moves with velocity v_R and movement angle θ_o away from the *X* axes into the new position $P(x_p, y_p)$

In case of the robot moves from the initial position $P_0=[x_0, y_0, \theta_0]^T$ at time t to the next position at $t+dt$, the trajectory is a circular arc with radius ρ, as shown in Fig. 2.8. Therefore the coordinate transformation of the new coordinate system is defined as follow

$$\begin{aligned} x' &= x\cos\theta_0 + y\sin\theta_0 - x_0\cos\theta_0 - y_0\sin\theta_0, \\ y' &= -x\sin\theta_0 + y\cos\theta_0 + x_0\sin\theta_0 - y_0\cos\theta_0. \end{aligned} \tag{2.9}$$

Whereas the inverse transformation is

$$\begin{aligned} x &= x'\cos\theta_0 - y'\sin\theta_0 + x_0, \\ y &= x'\sin\theta_0 + y'\cos\theta_0 + y_0. \end{aligned} \tag{2.10}$$

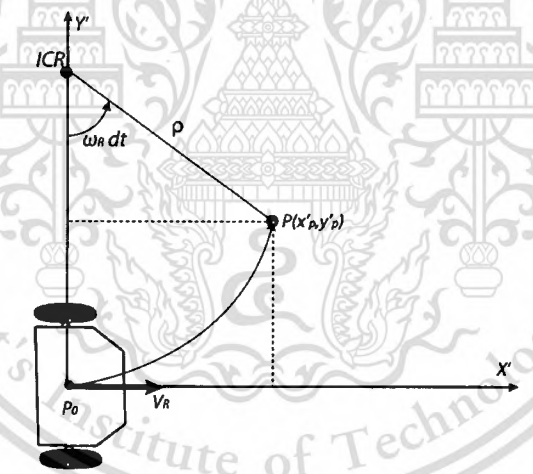


Fig. 2.9 The circular arc around the instantaneous center of rotation (IRC)

The initial position P_0 of robot is located in the origin of the transformed coordinate system as shown in Fig. 2.9. A new position after time interval dt is

$$\begin{aligned} x' &= \rho\sin(\omega_R dt), \\ y' &= \rho(1-\cos(\omega_R dt)). \end{aligned} \tag{2.11}$$

Applying the inverse coordinate transformation yields

$$\begin{aligned} x &= \frac{v_R}{\omega_R}(\sin(\omega_R dt)\cos\theta_0 - (1-\cos(\omega_R dt)\sin\theta_0)) + x_0\,, \\ y &= \frac{v_R}{\omega_R}(\sin(\omega_R dt)\sin\theta_0 + (1-\cos(\omega_R dt)\cos\theta_0)) + y_0\,. \end{aligned} \tag{2.12}$$

And

$$\theta = \omega_R dt + \omega_0. \tag{2.13}$$

For $\omega_R \to 0$ the new position is calculated from (2.11) by using 1'Hospital's rule

$$\begin{aligned} x' &= v_R dt, \\ y' &= 0. \end{aligned} \tag{2.14}$$

Applying the inverse coordinate transformation yields

$$\begin{aligned} x &= v_R dt\cos\theta_0 + x_0, \\ y &= v_R dt\sin\theta_0 + y_0\,, \end{aligned} \tag{2.15}$$

And

$$\theta = \theta_0. \tag{2.16}$$

Dividing both sides of Eqs. (2.15) and (2.16) by *dt*, but ignore x_0 and y_0 to get the sifting at *dt*. If *dt* tends to zero, then finally yield

$$\begin{aligned} x' &= v_R\cos\theta_0, \\ y' &= v_R\sin\theta_0, \end{aligned} \tag{2.17}$$

and

$$\theta' = \omega_R. \tag{2.18}$$

2.3.3 Embedded Chaotic Mobile Robot

Dynamic systems in chaos and their implementations have been studied intensively [10-15]. The implementations have been boarden to many fields such as encription [10, 11], secure communication [12, 13], and robotics. In the robotic field, a study of mobile robot attracted attention to many researchers is to explore chaotic dynamic systems ranging from laboratory to commercial scales [14]. Changkyu, *et al.* [15] study on path-planning algorithms for stationary obstacle avoidance implement the chaotic dynamics by using a neuron model. Generally robot designers use a standard map to generate mobile robot trajectory [16]. The chaotic mobile robot has advantage from non-repeatability in term of trajectories of the chaotic dynamics to accomplish area coverage [40]. Other implementations of chaotic dynamics for searching and patrol can be seen in [18, 19]. To generate chaotic behaviors, most chaotic robots are programmed by chaotic equations similar to the initial work of Nakamura and Sekiguchi [21].

2.3.3.1 Single Chaotic Mobile Robot

Single chaotic mobile robots with various implementations and applications have attracted much attention from researchers. To understand its concept, one can start from the work of Nakamura & Sekiguchi [21]. The work in [26] presents a method to control the motion of a mobile robot based on a chaotic system. The chaotic equation is embedded into the controller of the mobile robot to drive the direction of robot's motion. It has been proved that a chaotic system can guarantee the robot behaved chaotically. The robot can explore all workspace that given due to its chaotic behavior.

Let us start from the three-dimensional chaotic equation of Lorenz. Chaotic system is part of a dynamical system, and thus it can be represented by an ODE as mentioned in Eq. (2.1). Lorenz's system is stated below

$$\begin{aligned}\dot{L_1} &= \sigma(L_2 - L_1)\\ \dot{L_2} &= L_1(\rho - L_3) - L_2,\\ \dot{L_3} &= L_1 L_2 - \beta L_3\end{aligned} \tag{2.9}$$

where L_1, L_2, and L_3 are state variables regarding to Lorenz attractor and σ, ρ, and β are coefficients or constants. To achieve the chaotic strange attractor, one can set initial conditions $L_1 = 10, L_2 = 11,$ and $L_3 = 10;$ and set coefficients $\sigma = 10, \beta = \frac{8}{3}, and\ \rho$=28. The Lorenz's attractor can be seen in Fig.2.3.

A two-wheeled mobile robot can be modeled mathematically by adopting Eqs. (2.17) and (2.18), then the state equation of the mobile robot is able to write as follows:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}v\cos\theta & 0\\ v\sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}, \tag{2.20}$$

where v [*m/s*] is the linear velocity of the robot and ω [*rad/s*] represents the angular velocity. Whereas (x [*m*], y [*m*]) is the position of the robot on the workspace and θ [rad] defines the angle of the robot.

Chaotic behavior of mobile robot can be achieved by embedding the chaotic equation in Eq. (2.19) into the mobile robot equation in Eq. (2.20). Thus, the state equation of chaotic mobile robot becomes

$$\begin{bmatrix}\dot{L_1}\\ \dot{L_2}\\ \dot{L_3}\\ \dot{x_L}\\ \dot{y_L}\end{bmatrix} = \begin{bmatrix}\sigma(L_2 - L_1)\\ L_1(\rho - L_3) - L_2\\ L_1 L_2 - \beta L_3\\ v\cos L_3\\ v\sin L_1\end{bmatrix}. \tag{2.21}$$

In the numerical simulation, Eq. (2.21) is treated as the dynamical system of the ODE by using the fourth-order Runge-Kutta integragor [61]. Figure 2.10 shows the

trajectory of chaotic mobile robot which is generated by Eq. (2.21) along 83 *mins* in a $2 \times 2\ m^2$ workspace with constant velocity.

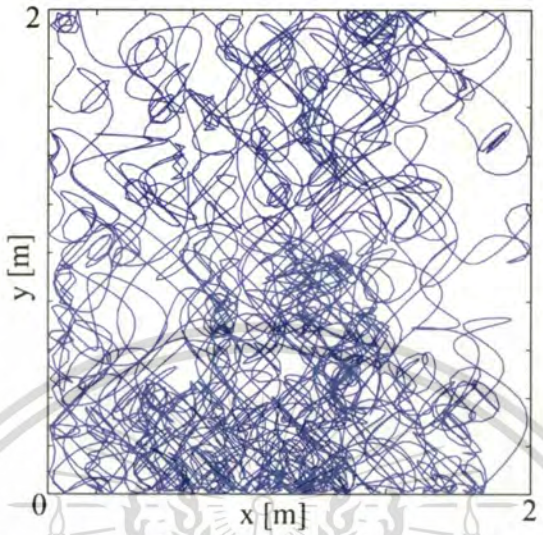


Fig. 2.10 The trajectory of chaotic mobile robot generated with embedding Lorenz system

In the case of real experimental, one can run Eq. (2.21) with the similar method of the numerical simulation into the microcontroller. For each iteration, the solution of (x_L, y_L) becomes the next position of the robot. It needs additional method such as dead reckoning to plan the path.

2.3.3.2 Complex Embedded Chaotic Mobile Robot

Complex networks have attracted attention from researchers around the world due to potential applications in the real-world such as social network, Internet, food-web network, neural network and so on [62]. Those networks are characterized by huge elements, borderless structure, and dynamic change. Among the complex systems, chaos is one of phenomena appeared regularly when deterministic systems exhibit non-periodic behaviors that depends sensitively on the initial conditions. The number of agents in the colony or network can behave chaotically with proper interaction non-linearly. Moreover, even a network with four agents can exhibit chaotic behaviors [23, 63].

This fact has inspired Rosyid & Sooraksa [64] simulate complex embedded chaotic mobile robot. The idea is adopted from Sprott's equation of the minimum chaotic network in [23] which is considered as a model of complex network system. The system can exhibit a dynamics including chaos with four agents involved. The state equation is written as follow:

$$\begin{aligned}\dot{x}_1 &= -bx_1 + \tanh(x_4 - x_2)\\ \dot{x}_2 &= -bx_2 + \tanh(x_1 + x_4)\\ \dot{x}_3 &= -bx_3 + \tanh(x_1 + x_2 - x_4)\\ \dot{x}_4 &= -bx_4 + \tanh(x_3 - x_2)\end{aligned} \tag{2.22}$$

where b is the damping coefficient, decreasing this value is analogous to increasing the interaction among the agents. This system exhibits chaos at $b=0.043$ and initial conditions are set to x_1=1.2, x_2=0.4, x_3=1.2 and x_4=-1.0. Figure 2.11 shows the chaotic strange attractor of Sprott's system.

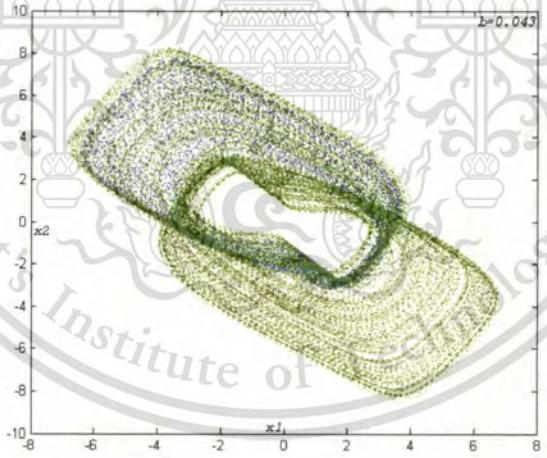


Fig. 2.11 Chaotic strange attractor of Sprott's system

In order to realize complex embedded chaotic mobile robots, we insert dynamic weight of interaction among the agents (robot) into the original Sprott's system. Figure 2.12 illustrates robots' interaction by using a directed graph based on the dynamic weight of interaction. The structure of the directed graph that shown in Fig. 2.12 is derived from modified Sprott's system which is written below;

$$\begin{aligned}\dot{s}_1 &= -bs_1 + \tanh(w_{14}s_4 - \mathrm{w}_{12}s_2)\\ \dot{s}_2 &= -bs_2 + \tanh(\mathrm{w}_{21}s_1 + w_{24}s_4)\\ \dot{s}_3 &= -bs_3 + \tanh(w_{31}s_1 + w_{32}s_2 - \mathrm{w}_{34}s_4)'\\ \dot{s}_4 &= -bs_4 + \tanh(\mathrm{w}_{43}s_3 - w_{42}s_2)\end{aligned} \tag{2.23}$$

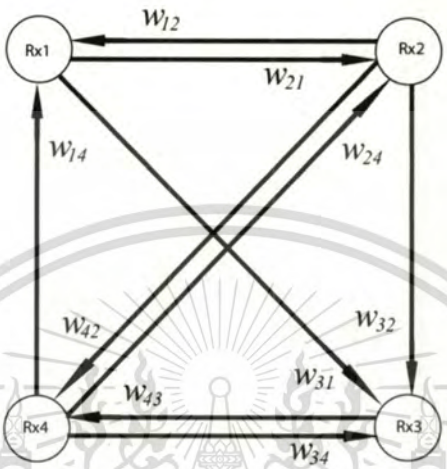


Fig. 2.12 Directed graph of the dynamic weight of complex embedded chaotic mobile robots

where w_{ij} is the weight of robot's interaction. According to the *audible-inaudible* scenario, w_{ij} is obeyed the inverse square law of distance between robots i^{th} and j^{th}. Let define r_{ij} as Euclidean distance between robots i^{th} and j^{th}. If robots i^{th} and j^{th} move apart and the value of r_{ij} is greater than a threshold k_j, then the value of $w_{ij}=r_j^{-2}$ (1 is otherwise). This condition can be described by a threshold operation as written below,

$$W = \delta_T(r_{ij}),\ w_{ij}\begin{cases}1 & r_{ij} \le T\\ r_{ij}^{-2} & r_{ij} > T\end{cases}, for\ T = k_j, \tag{2.24}$$

where W is a weight matrix, w_{ij} is the element of W, and T is threshold value with the default 1.

The integration of modified Sprott's equation and the mobile robot equation is adopted from the same method as at the single chaotic mobile robot. Embedding Eq. (2.23) into Eq. (2.20) yields the state equation of the complex chaotic mobile robot as follow

$$\begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \dot{s}_3 \\ \dot{s}_4 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_3 \\ \dot{y}_3 \\ \dot{x}_4 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} -bs_1 + \tanh(w_{14}s_4 - w_{12}s_2) \\ -bs_2 + \tanh(w_{21}s_1 + w_{24}s_4) \\ -bs_3 + \tanh(w_{31}s_1 + w_{32}s_2 - \mathrm{w}_{34}s_4) \\ -bs_4 + \tanh(\mathrm{w}_{43}s_3 - w_{42}s_2) \\ v\cos s_1 \\ v\sin s_1 \\ v\cos s_2 \\ v\sin s_2 \\ v\cos s_3 \\ v\sin s_3 \\ v\cos s_4 \\ v\sin s_4 \end{bmatrix}, \tag{2.25}$$

where (*xi,yi*) is position of a robot in the Cartesian coordinate.

Computer simulation shows that the chaotic behavior of robot network is preserved as long as interactions among robots are high or equals to 1 [70]. But, if the robot network does not satisfy this condition, the chaotic behavior will disappear as the weight of each robot obeys the inverse square law. Figures 2.13 and 2.14 are results of the numerical simulation that show the behaviors of each robot and distance among robots, respectively. Further investigation inspired by this early work will be presented later in next chapters.

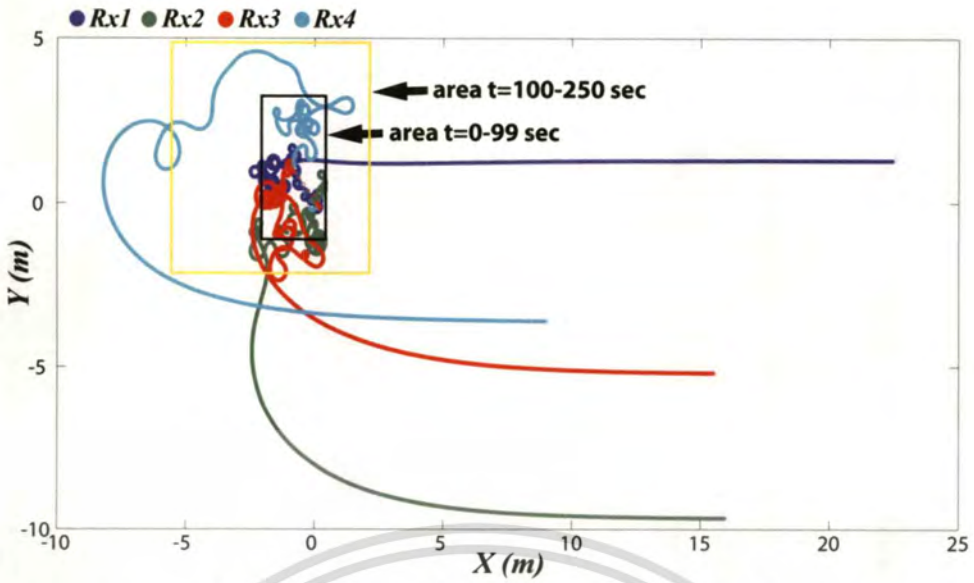


Fig. 2.13 Behaviors of complex chaotic mobile robots on the X-Y plane. Within a first 99 sec the chaotic behaviors are preserved by strong interactions. But after that the robot network lost the weight of its interaction, and the chaotic behaviors cannot be longer preserved

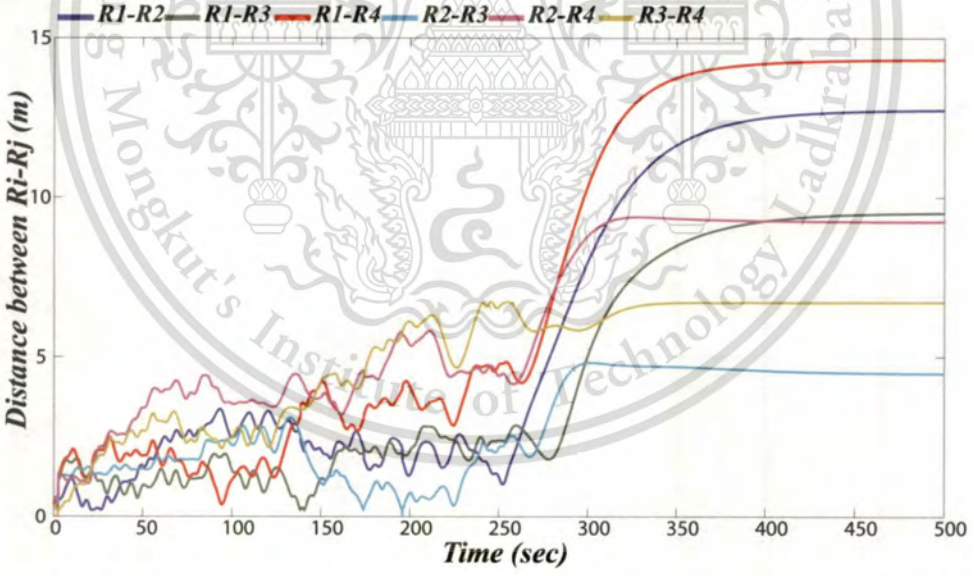


Fig. 2.14 Distance among robots at various time, each curve represents distance between Robots *i* and *j*

Chapter 3

Analysis on Behaviors of Sequential Patterns of Botnet Attacks

3.1 Introduction

The Internet is a kind of complex network that includes of huge number of networks of computers. The botnet itself is a sub of the Internet that comprises compromised computers by botmaster. The botnet is established by botmaster with spreading malware through the Internet in a particular pattern. Therefore, the investigation of the behavior of malware gives benefits to defend against the botnet's threats.

This chapter discusses the discovery of new frequent sequential attack patterns of malware that sent by botnets. In this research, 94 independent honeypots that have observed malware traffic on the Japanese tier-1 backbone have been investigated. The observations were coordinated by the Cyber Clean Center (CCC) - https://www.ccc.go.jp. The CCC Datasets for 2009 and 2010 comprise the access logs of attacks that have been collected by honeypots since May 1, 2008 through May 31, 2010.

Due to the featured confidentiality of resources, the research has been conducted only in the limited information provided by the CCC. Accessible information are CCC Datasets that consists of Timestamps, Honeypot ID, Source/Distenation port number, Source IP address, Source port number, Hash value (SHA1), Malware name, and Malware file name.

This chapter starts with an explanation of the *PrefixSpan* algorithm, and then continued to mining the dataset. We next define the naming pattern, length of pattern and type of pattern. After that, classifying the sequential attacks pattern

based on IP address and timestamps are conducted. Afterwards, analysis of the behavior is presented in some charts. Further analysis of behavior based on entropy is also discussed.

This chapter reveals the sequential behavior of malware spreading through the Internet to establish the botnet. On the other hand, the complex network from the dynamic system point of view has a wide range of behaviors from dead fixed point, periodic sequential, until route to chaos [23, 63]. Accordingly, after the botnet in the cyberspace is investigated in this chapter, the next chapter will investigates behaviors of complex robotic network in the real space.

3.2 The *PrefixSpan* Algorithm

The investigation of CCC Datasets is based on a data mining algorithm, namely the *PrefixSpan* algorithm of Pei et al. [65]. *PrefixSpan* is applied in this research because it is an algorithm for the efficient mining of sequential patterns in a huge dataset that avoids the requirement to construct candidate sets and makes low memory demands[66], in comparison with a priori-like algorithm [67, 68].

Sequential pattern mining is a method for discovering subsequence patterns in a database. This method was introduced by Agrawal and Srikant [68] and describe as follows. *Given a set of sequences, where each sequence comprises a list of elements and each element comprises a set of items, and given a user-specified minimum support threshold as a condition, sequential pattern mining aims to find all frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is greater than or equal to the minimum support threshold.* A sequential pattern-mining method called Prefix-projected Sequential Pattern Mining (*PrefixSpan*), which discovers frequent subsequences as patterns in a sequence database, was initially proposed by Pei et al. [65]

Let a_i, b_j be items and let α_i, β_j be sequences of items, where $\alpha = \langle a_1 a_2 \cdots a_n \rangle$ and $\beta = \langle b_1 b_2 \cdots b_n \rangle$. Then α is a **subsequence** of β, denoted by $\alpha \sqsubseteq \beta$, if and only if there exist integers $j_1, j_2, \cdots, j_n$ such that $1 \leq j_1 < j_2 < \cdots < j_n \leq m$ and $a_1 = b_{j_1}, a_2 = b_{j_2}, \cdots, a_n = b_{j_n}$. A **sequence database** *S* is a set of tuples $\langle sid, s \rangle$, where *sid* is a **sequence_id** and *s* is a **sequence**. The **support** of a sequence α in a database *S* is the number of tuples in the database containing α, i.e.,$support(\alpha) = |\{\langle sid, s \rangle | \langle sid, s \rangle \in S, \alpha \sqsubseteq s\}|$. Given a positive integer *min_sup* as a support threshold, a sequence α is called a **frequent sequential pattern** in database *S* if the sequence is contained by at least *min_sup* tuples in the database, i.e.,$support(\alpha) \geq \min\_sup$. The number of items in a sequence is called the **length** of the sequence, with a sequential pattern of length ℓ being called an **ℓ-pattern**.

Let α and β be sequences $\langle a_1 \cdots a_n \rangle$ and $\langle b_1 \cdots b_n \rangle$, respectively. In terms of the *PrefixSpan* algorithm, we identify the following.

1) **Prefix and Postfix**: sequence α is a prefix β of if and only if $a_i = b_i$ for $i = 1, \cdots, m$. For example, $\langle a\ a\ b\ c \rangle$ is prefix of $\langle a\ a\ b\ c\ d\ d\ a\ b \rangle$. The sequence following a prefix is its postfix, $\langle d\ d\ a\ b \rangle$ in this case.
2) **Projection**: Let α, β, γ be sequences such that $\beta \sqsubseteq \alpha, \gamma \sqsubseteq \alpha$. Sequence γ is a **β-projection** of α if and only if (1) β is a prefix of γ, and (2) there exists no longer subsequence of α such that β is its prefix. For example, the **c**-projection of $\langle a\ a\ b\ c\ d\ c\ d\ a\ b \rangle$ is $\langle d\ c\ d\ a\ b \rangle$.

Table 3.1 A sequence database

Sequence ID	Sequence
100	PE WO TR
200	PE TR WO
300	BK PE TR TS WO
400	TS PE PE TR WO BK
500	PE WO TR WO

As an example, consider the sequence database *S* in Table 3.1. If the user specifies *min_sup*=2, then the sequential patterns in *S* can be mined by the *PrefixSpan* method in the following steps.

1) **Step 1: Find 1-pattern sequences.**

   Scan the database *S* once to discover all frequent items in the sequences. These are $\langle PE\rangle: 5, \langle WO\rangle: 5, \langle TR\rangle: 5, \langle BK\rangle: 2$, and $\langle TS\rangle: 2$, where $\langle pattern\rangle: count$ is the pair of the pattern and the support count.

2) **Step 2: Distribute the search space.**

   The *projected database* can be distributed across the following five subsets according to the five prefixes that resulted from Step 1: (1) those having prefix $\langle PE\rangle, \cdots$, and (5) those having prefix $\langle TS\rangle$.

3) **Step 3: Find subsets of sequential patterns.**

   These can be mined by constructing the five corresponding projected databases and repeating the process with each of them recursively.

Table 3.2 Sequential patterns

Prefix	Projected Databases	Sequential Pattern
$\langle PE\rangle$	$\langle WO\ TR\rangle, \langle TR\ WO\rangle$ $\langle TR\ TS\ WO\rangle, \langle PE\ TR\ WO\ BK\rangle$ $\langle WO\ TR\ WO\rangle$	$\langle PE\rangle: 5$ $\langle PE\ TR\rangle: 5$ $\langle PE\ TR\ WO\rangle: 4$ $\langle PE\ WO\rangle: 5$ $\langle PE\ WO\ TR\rangle: 2$
$\langle WO\rangle$	$\langle TR\rangle, \langle BK\rangle$	$\langle WO\rangle: 5$ $\langle WO\ TR\rangle: 2$
$\langle TR\rangle$	$\langle WO\rangle, \langle TS\ WO\rangle, \langle WO\ BK\rangle, \langle WO\rangle$	$\langle TR\rangle: 5$ $\langle TR\ WO\rangle: 4$
$\langle BK\rangle$	$\langle PE\ TR\ TS\ WO\rangle$	$\langle BK\rangle: 2$
$\langle TS\rangle$	$\langle WO\rangle, \langle PE\ PE\ TR\ WO\ BK\rangle$	$\langle TS\rangle: 2$ $\langle TS\ WO\rangle: 2$

Starting from prefix $\langle PE \rangle$, we can make a $\langle PE \rangle$-projected database that consists of five postfix sequences: $\langle WO\ TR \rangle$, $\langle TR\ WO \rangle$, $\langle TR\ TS\ WO \rangle$, $\langle PE\ TR\ WO\ BK \rangle$, and $\langle WO\ TR\ WO \rangle$. In a recursive procedure, we back to Step 1 by scanning the $\langle PE \rangle$-projected database once, bringing all 2-pattern sequences having prefix $\langle PE \rangle$, those are $\langle PE\ WO \rangle$: 5 and $\langle PE\ TR \rangle$: 5. Next, the $\langle PE \rangle$-projected database is broken down into two subsets respect to its two prefixes $\langle PE\ WO \rangle$ and $\langle PE\ TR \rangle$. Each generated projected database is then mined recursively. With prefix $\langle PE\ WO \rangle$ having three postfix sequences $\langle TR \rangle, \langle BK \rangle$, and $\langle TR\ WO \rangle$, mining these sequences results in the sequential pattern $\langle PE\ WO\ TR \rangle$, which cannot be explored further because its support count is less than *min_sup*. With prefix $\langle PE\ TR \rangle$ carrying four postfix sequences $\langle WO \rangle$, $\langle TS\ WO \rangle$, $\langle WO\ BK \rangle$, and $\langle WO \rangle$, we get 3-pattern output $\langle PE\ TR\ WO \rangle$: 4. Eventually, Table 3.2 shows the complete projected database and the sequential patterns.

3.3 Sequential Pattern in Botnet Attacks

Malware footprints of several botnets spread across the network can be captured by honeypots. The study of datasets harvested from honeypots gives us benefits to be able to investigate further behavior of botnets attacks. Because botnets use systematic attack methods, the sequences of malware downloaded by honeypots have particular forms of coordinated pattern.

PrefixSpan algorithm is implemented in this research to reveal sequential behaviors of coordinated pattern of botnet attacks. The dataset is filtered to get the sequence database of malware name with respect to its timestamps. Once the sequence database is built, the mining process can be run. The mining dataset gives the result in the form of a list of sequential patterns of malware name. Further investigations of the mining result deliver valuable information such as classification of patterns based on feature of the sequence, IP address, timestamps, and duration of attack.

3.3.1 Preprocessing Data

The CCC Dataset consists of the yearlong access logs of botnet attacks recorded by 94 independent of honeypots. Each honeypot reboots every 20 minutes, and between each reboot, the honeypot records every inbound packet in an access log. The 20-minute period is called a time slot, or simply a "slot". Figure 3.1 shows the example of CCC Dataset.

```
2008-05-01 02:20:56,honey001,1035,x.x.x.x,80,TCP,f8845399071e0c480bfc3202bf4e74eaa45e648b,BKDR_AGENT.ANHZ,C:\WINDOWS\system32\wffkjz.exe
2008-05-01 02:26:28,honey001,1027,x.x.x.x,54269,TCP,72a85576f7f833feeb603cd6539a172f4664cd2a,TROJ_POEBOT.AGU,C:\WINDOWS\system32\iltgorcz.exe
2008-05-01 02:27:24,honey001,1028,x.x.x.x,69,UDP,9abb0afaf9a06cda355f052203d2d1e2a7686c38,BKDR_RBOT.FTV,C:\WINDOWS\system32\wbem\winscrvs.exe
---------------------------
2008-05-01 02:42:30,honey001,1030,x.x.x.x,34486,TCP,7eb8cbee42c44e7aa696a5fb4d598a05d9f976cb,PE_VIRUT.AV,C:\WINDOWS\system32\edgyt.exe
2008-05-01 02:43:31,honey001,1036,x.x.x.x,80,TCP,34f4ff4acb18802170a939ae42dcd5ee0eeccda4,TROJ_MATCASH.AO,C:\WINDOWS\Temp\VRT2.tmp
2008-05-01 02:52:34,honey001,1039,x.x.x.x,34486,TCP,7eb8cbee42c44e7aa696a5fb4d598a05d9f976cb,PE_VIRUT.AV,C:\WINDOWS\system32\smwtmw.exe
2008-05-01 02:53:32,honey001,1042,x.x.x.x,80,TCP,34f4ff4acb18802170a939ae42dcd5ee0eeccda4,TROJ_MATCASH.AO,C:\WINDOWS\Temp\VRT1.tmp
2008-05-01 02:54:11,honey001,1044,x.x.x.x,38641,TCP,02970a20c8ae3307772570a29671a7344aeb9ba3,BKDR_VANBOT.MF,C:\WINDOWS\system32\algs.exe
2008-05-01 02:54:17,honey001,1048,x.x.x.x,80,TCP,d7b9b9b10d9f7d2c961365b72e189eb95a9f03f8,UNKNOWN,C:\WINDOWS\system32\qyqwpcj.exe
```

Fig. 3.1 The snapshot of CCC dataset where is the dash line is divider for each 20 minutes, and each line contains attributes that recorded which is separated by comma. Due to the confidential information, the IP addresses are masked with *x.x.x.x*

In this research, the *PrefixSpan* is implemented to discover frequent attack patterns based on the sequences of malware downloaded by honeypots. Consequently, we need a suitable form of database that fit the requirement of the *PrefixSpan* algorithm. The CCC Dataset is filtered to get malware names in a sequence and write down into the form of a text file. The text file comprises *lines*, with each line being a sequence of malware names. The term *line* is interchangeable with *slot*, reflecting the terminology for honeypots used later in our discussion. The timestamps for the downloaded malware determine the order of the malware within the slot. An example of preprocessed data is given in Table 3.3.

Table 3.3 Example of preprocessing data for a sequence database

Slot	Sequence of Malware Names
0	TROJ_SYSTEMHI.BQ
1	KDR_AGENT.ANHZ UNKNOWN TROJ_SYSTEMHI.BQ BKDR_AGENT.ANHZ
2	PE_BOBEX.AH
⋮	⋮
15323	PE_VIRUT.AV TROJ_IRCBRUTE.BW WORM_AUTORUN.CZU
15324	UNKNOWN PE_VIRUT.AV PE_VIRUT.AV WORM_AUTORUN.CZU

3.3.2 The Selection of the Length of Pattern (ℓ-patern)

A botnet assault is associated with a highly organized, coordinated, and systematic strategy. Consequently, the sequence of malware downloaded by the honeypots must be in a particular form of coordinated pattern. To reveal accurately the coordinated attack patterns, we use the number of malware items in a pattern, i.e., the length of the sequence/pattern. The question is what length of pattern (ℓ-**pattern**) of the sequential pattern to be investigated further? To answer this question, we make a simple statistical analysis against the CCC Dataset for helping to choose the reasonable ℓ-**pattern.**

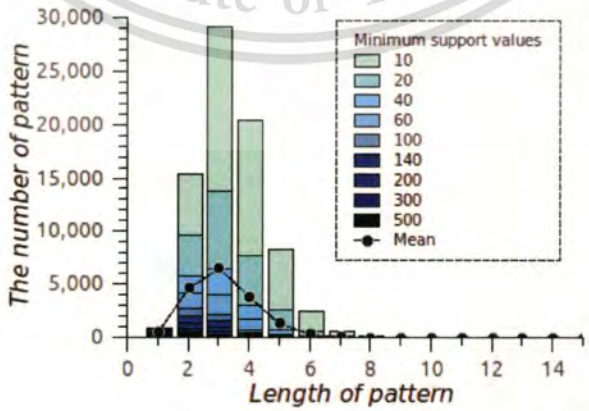


Fig. 3.2 The distribution of the number of pattern that mined as the length of pattern is varied

Figure 3.2 shows the distribution of the lengths of sequences in malware as the minimum support values are varied from 10 to 500, where the *Y*-axes gives the number of patterns as the function of the length of pattern *X*-axes that are varied from 1 to 14. Notice that the number of patterns resulted from the mining dataset with the length pattern 1 are the same as the number of pattern in its original sequence database. The distribution of the lengths of sequences tends to be between length 2 and length 3 on the *X*-axes. These fundamental features are useful in tuning the parameters in the data mining, i.e., they become a reference point for selecting the sequence length (**ℓ-pattern**), and thus 3-pattern will be reasonable to be investigated deeply.

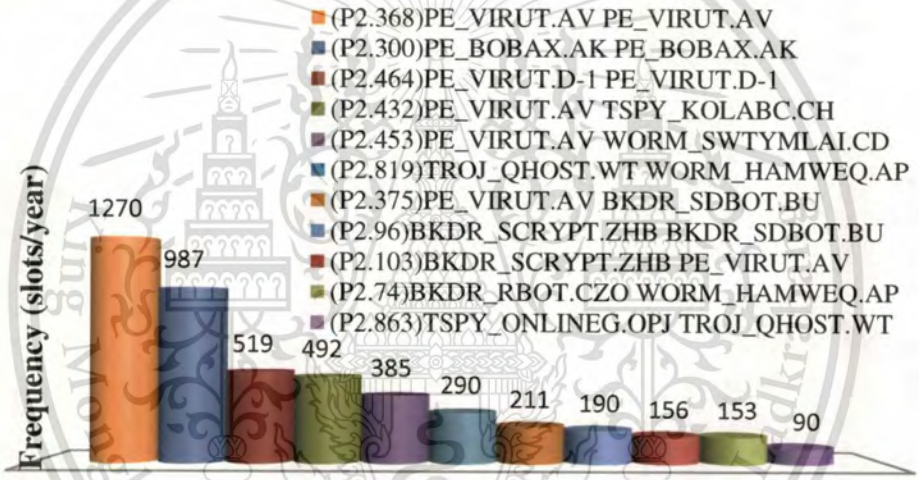
Mining of the CCC dataset using the *PrefixSpan* algorithm produces a list of the sequential attack patterns of malware. A particular slot is said to be infected slot if there exist a sequential pattern of malware in that slot. The list is sorted according to the number of slots that are infected by malware. The accuracy of the minimum support determines the number of sequential attack patterns discovered. For example, the minimum support is set to 1, if there exist a unique sequential pattern of the whole dataset, and then it will put into account. Thus, *PrefixSpan* will result a huge number of sequential patterns. This situation will not efficient for further investigation. Otherwise, if the minimum support is set to the high value, *PrefixSpan* will give a small number of sequential patterns. Consequently, it decreases the accuracy of the farther analysis.

In this investigation, we select the minimum support for the sequential attack 2-patterns using the assumption that each honeypot reboots every 20 minutes, giving 72 slots per honeypot per day. If there are intensive attacks in a certain day, the number of slots infected will be less than or equal to 72. Let us suppose that 70 is reasonable as the minimum support for discovering sequential attack 2-patterns. Referring to the trend of the distribution of the lengths of sequences shown in Fig.

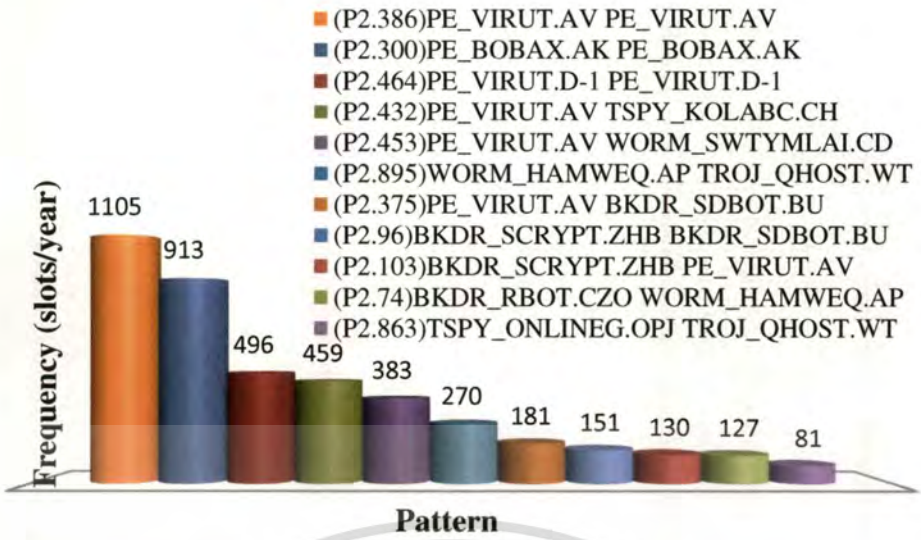
3.2, we choose 30 as the minimum support for 3-patterns, i.e., 40% of the minimum support for the sequential attack 2-patterns.

3.3.3 The Definition of Sequential Attack Patterns

The sequential attack patterns identified are indexed into the form $P_{x,y}$ to simplify the naming of patterns, where x is the sequence length of the pattern and y is its serial number in the naturally ordered list. As an example of index naming, pattern $P_{3.1203}$ is a sequential attack pattern with a sequence length of 3 and occupying line 1203 in the list.



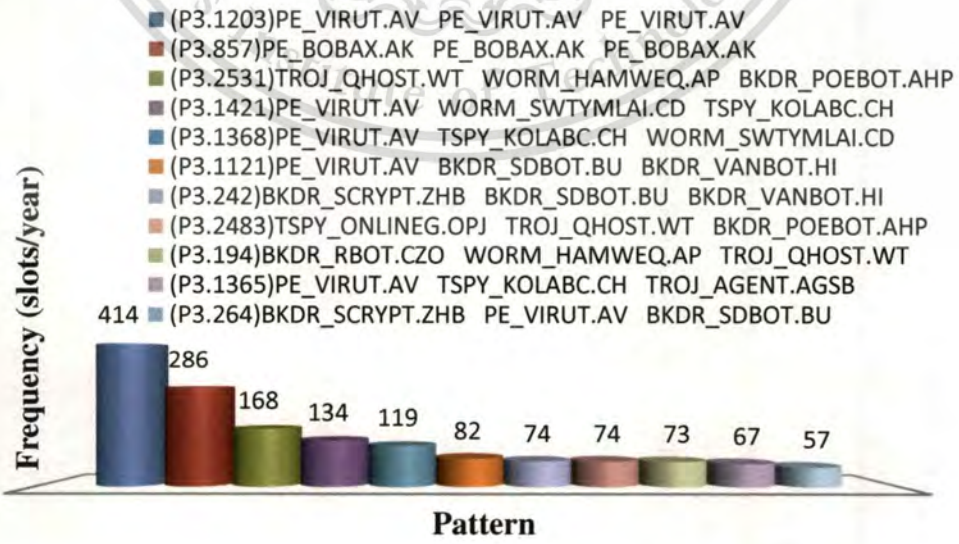
(a) Honeypot 1



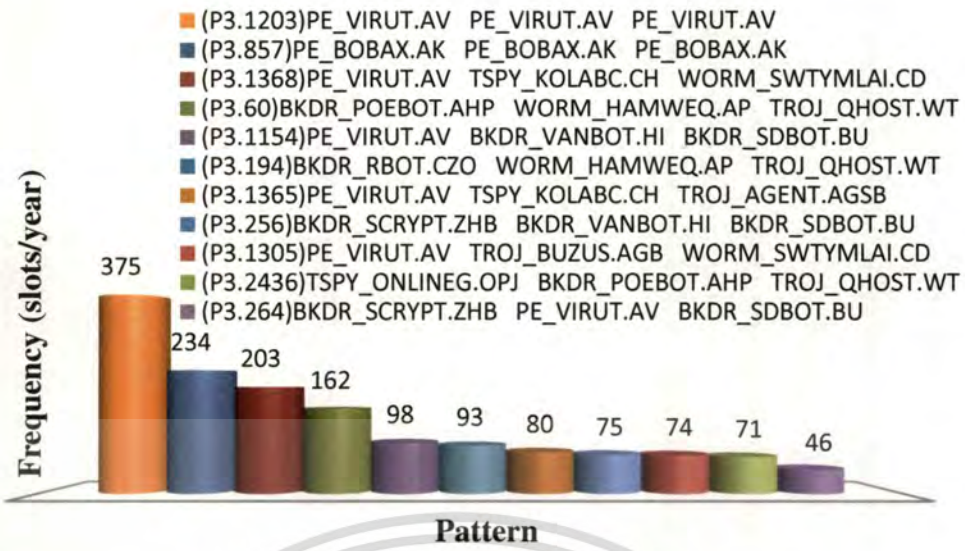
(b) Honeypot 2

Fig. 3.3 Sequential attack 2-pattern of malware: (a) at the honeypot 1 and (b) at the honeypot 2

Based on the form of the malware sequence, the mining results can be classified into two categories, *duplicate* and *nonduplicate*, where a duplicate pattern, unlike a nonduplicate pattern, it has the same malware appearing more than once in it. For example, Fig. 3.3(a) shows patterns $P_{2.386}$ and $P_{2.300}$, and Fig. 3.4(a) shows patterns $P_{3.1203}$ and $P_{3.857}$. These patterns include the duplicated malware *PE_VIRUT.AV* and *PE_BOBAX.AK*.



(a) Honeypot 1



(b) Honeypot 2

Fig. 3.4 Sequential attack 3-pattern of malware: (a) at the honeypot 1 and (b) at the honeypot 2

The behaviors of the sequential attack patterns for all honeypots seem to be similar. Therefore, we investigate just two of the 94 sample honeypots further, namely Honeypot 1 running under Windows XP+SP1 and Honeypot 2 running under Windows 2000. These two behaviors will be sufficient to represent the behavior of the sequential attack patterns in general.

3.3.4 The Sequential 2-Pattern and 3-Pattern of Botnet Attacks

The results of mining the sequential attack 2-patterns for these two honeypots are depicted in the column-bars diagrams for Honeypots 1 and 2, respectively, as shown in Figs 3.3 (a) and (b), where the *X*-axes is the pattern name and the *Y*-axes is the download frequency (in *slots/year*).

According to the study, there is a significant relationship between these two honeypots in terms of the order in the list of the sequential attack patterns and the download frequency over a year. The five top-ranked sequential attack 2-patterns of

malware from both honeypots have the same sequential pattern. Slightly differences in the number of infected slots are found in the other places at 2-patterns of sequential attacks, as shown in Figs. 3.3 (a) and (b). For example, the same pattern of $P_{2,453}$ for Honeypots 1 and 2 have very small differences (2 slots) in the download frequency which are recorded 385 and 383 *slots/year*, respectively. This fact indicates that there is no dependency of the Operating System in the botnets point of view. As we know that both of honeypots under investigation are the same Operating System vendor of Microsoft Windows. In the other words Windows XP+SP1and Windows 2000 have a common point of vulnerable.

As much as 169 and 118 patterns (including 29% and 26% of nonduplicate patterns) of the sequential attack 3-pattern successfully revealed by mining datasets of honeypot #1 and #2, respectively. Sequential attacks 3-pattern for both honeypots are detailed in the column-bars chart as shown in Fig. 3.4. Duplicate patterns that consist of PE_VIRUT.AV and PE_BOBAX.AK dominate the top rankings of the lists for both sequential 2-patterns and 3-patterns. This duplication implies that the malware has successfully infected the honeypot more than once in a single slot. These facts may be regarded as an indication that these are the most common malware to have been employed by a botnet system.

3.3.5 Attack Pattern Based on IP Address and Timestamps

Through the Internet the botnet spreads malware to bots. A vigilance and sensitivity against threats of botnet attacks can be established by understanding the behavior of the spread of malware from its source IP address and malware downloaded timestamps. Thus, the investigation of the source IP address and malware downloaded timestamps of the botnet attacks become very important.

Firstly, several groups of source IP address and malware downloaded timestamps are built for classifying sequential attacks 3-pattern of malware. The naming for classifying the sequential attack 3-pattern based on source IP address is

designed to use symbols $A_1,A_2,...,A_5$ for representing *IP pattern code*, and then *IP pattern* of the sequence of source IP address of malware attacks are defined by using symbols S_1,S_2 and S_3 associated to the source IP address. It is summarized in Table 3.4. Another classification is based on the malware downloaded timestamps. The naming is similar with source IP address. Symbols E_1, $E_2,...,E_4$ represent the time pattern code, meanwhile the timelines of honeypots downloading the malware are marked with symbols T_1,T_2 and T_3 as shown in Table 3.5. For example, if pattern $P_{3,242}$ is of type A_3E_3, as shown in Table 3.6, the first and third malware items are downloaded from the same source IP address (A_3), and the second and third malware items are downloaded at the same time (E_3).

Table 3.4 Naming of attack patterns based on the source IP address

IP Pattern Code	IP Pattern		
A_1	S_1	S_1	S_1
A_2	S_1	S_1	S_2
A_3	S_1	S_2	S_1
A_4	S_1	S_2	S_2
A_5	S_1	S_2	S_3

Table 3.5 Naming of attack patterns based on malware download timestamps

Time Pattern Code	Time Pattern		
E_1	T_1	T_1	T_1
E_2	T_1	T_1	T_2
E_3	T_1	T_2	T_2
E_4	T_1	T_2	T_3

Table 3.6 List of the sequential attack 3-patterns for the malwares

#HN[1]	ID	Freq.	Sequential Attack Patterns	Avg[s]	SD[s]	Unique Host			Type	Gr.[2]
1	$P_{3.2351}$	168	TROJ_QHOST.WT WORM_HAMWEQ.AP BKDR_POEBOT.AHP	4.27	51.07	1	1	1	A_1E_1	A
	$P_{3.2483}$	74	TSPY_ONLINEG.OPJ TROJ_QHOST.WT BKDR_POEBOT.AHP	97.04	165.46	41	1	1	$A_4E_{1,3}$	A
	$P_{3.194}$	73	BKDR_RBOT.CZO WORM_HAMWEQ.AP TROJ_QHOST.WT	56.65	235.71	3	1	1	A_1E_1	A
2	$P_{3.60}$	162	BKDR_POEBOT.AHP WORM_HAMWEQ.AP TROJ_QHOST.WT	34.12	175.92	8	1	1	A_1E_1	A
	$P_{3.2436}$	93	TSPY_ONLINEG.OPJ BKDR_POEBOT.AHP TROJ_QHOST.WT	72.66	191.33	34	1	1	A_4E_3	A
	$P_{3.194}$	71	BKDR_RBOT.CZO WORM_HAMWEQ.AP TROJ_QHOST.WT	381.48	478.60	5	1	1	$A_1E_{1,3}$	A
1	$P_{3.1121}$	82	PE_VIRUT.AV BKDR_SDBOT.BU BKDR_VANBOT.HI	108.31	212.90	48	1	1	$A_3E_{1,3}$	B
	$P_{3.242}$	74	BKDR_SCRYPT.ZHB BKDR_SDBOT.BU BKDR_VANBOT.HI	732.12	422.57	11	1	1	$A_{3,5}E_3$	B
	$P_{3.264}$	57	BKDR_SCRYPT.ZHB PE_VIRUT.AV BKDR_SDBOT.BU	862.60	304.87	5	42	1	$A_5E_{3,4}$	B
2	$P_{3.1154}$	98	PE_VIRUT.AV BKDR_VANBOT.HI BKDR_SDBOT.BU	75.54	177.64	55	1	1	A_5E_3	B
	$P_{3.256}$	75	BKDR_SCRYPT.ZHB BKDR_VANBOT.HI BKDR_SDBOT.BU	821.86	326.30	6	2	1	$A_{2,5}E_3$	B
	$P_{3.264}$	46	BKDR_SCRYPT.ZHB PE_VIRUT.AV BKDR_SDBOT.BU	968.42	258.12	6	34	1	$A_5E_{3,4}$	B

[1] HN : Honeypot number

[2] Gr. : Group of attacks

Source IP addresses of malware play the important role in the investigation of botnet attacks. Some malware are sent by botnet through a unique source IP address and others from multiple of source IP address. The numbers of unique-host and IP-pattern types of the sequential attack 3-patterns are given in Table 3.6. Some of the sequential attack 3-patterns are of the single-source type, but others have two sources. Patterns in the top ranking are classified into two groups based on similarities in download times.

This study identifies two groups that are worthy of investigation, *A* and *B*, as shown in Table 3.6. The difference in the source-IP pattern types exist in the attacker groups *A* and *B*. The source-IP pattern types A_1, A_4 and E_1 are quite often utilized by the sequential attacks 3-pattern in Group *A*. The column of Unique Host shows patterns $P_{3.2351}$, $P_{3.194}$ and $P_{3.60}$ are downloading malware simultaneously only from a unique host, even though the first malware item is downloaded from a few different hosts. As shown in the *unique host* column in Table 3.6, the leftmost field is the host of the first malware, followed by the second and third.

Two patterns member of the group *A* shows a different behavior as represented by patterns $P_{3.2483}$ and $P_{3.2436}$. As shown in Table 3.6, *TSPY_ONLINEG.OPJ* is a common malware in the first order that composes of these patterns, and it does not appear in the other patterns in Group A. This malware comes from various unique of hosts by 41 and 34 number of unique hosts at Honeypot #1 and #2, respectively. In contrast, the second and third malware of $P_{3.2483}$ and $P_{3.2436}$ are downloaded from single unique host by average time interval 97.04 and 72.66 *second*, respectively.

The source-IP pattern types A_3, A_5 and E_3 are always matching for the sequential attack 3-pattern of Group *B*, as shown in Table 3.6. Generally, all malware items in the sequential patterns come from different hosts. Yet, it seems only the first malware item that comes from many unique of sources, whereas the second and third items are downloaded from a few unique of hosts. This involves patterns

$P_{3.1121}$, $P_{3.242}$, $P_{3.1154}$, and $P_{3.256}$. In Group *B*, pattern $P_{3.264}$ was downloaded by both honeypots. The first and third malware items in pattern $P_{3.264}$ have few unique hosts, but the second item *PE_VIRUT.AV* has downloaded from many different unique hosts by 42 and 43 at the honeypot #1 and #2, respectively, as shown in the *unique host* column of Table 3.6. This can be considered as evidence that the botnet employs a collaboration and coordination strategy to attack victims.

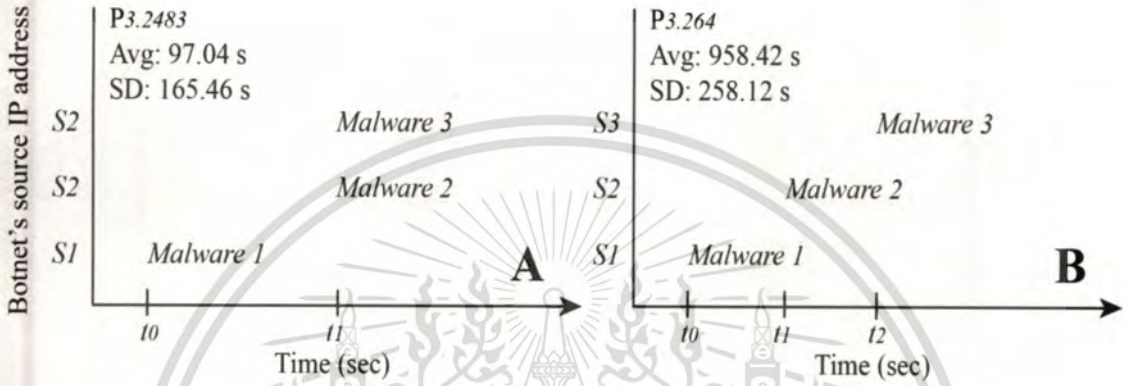
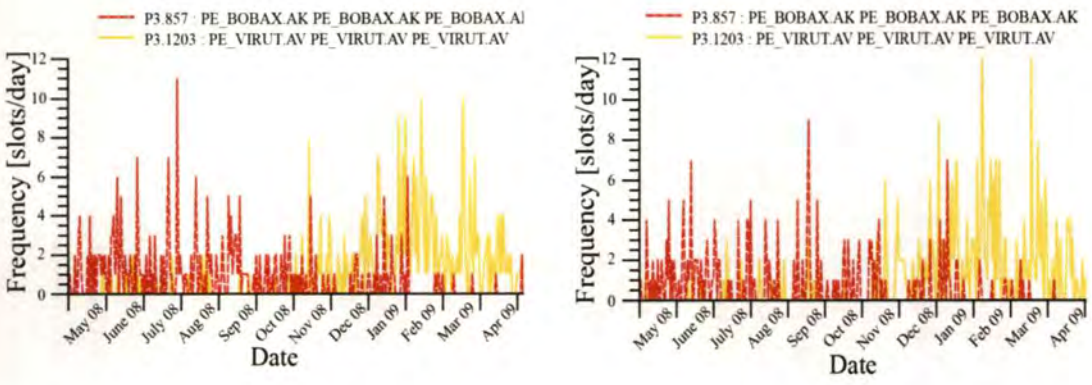


Fig. 3.5 Time charts for coordinated attacks made by the sequential attacks 3-pattern: (A) $P_{3.2483}$ belongs to IP pattern A_4 and time pattern E_3, (B) $P_{3.264}$ belongs to IP pattern A5 and time pattern E_4

Figure 3.5 shows time charts for coordinated attacks made by the sequential attack 3-patterns $P_{3.2483}$ and $P_{3.264}$. It illustrates how the coordinated attacks work. These are observed in terms of the source IP addresses and timestamps given in Tables 3.4 and 3.5. Consequently, naming and mapping the behaviors of sequential attack patterns may inform us about the spread of malware through the network, and lead us to identify and anticipate threats much earlier.



(a) Honeypot 1 (b) Honeypot 2

Fig. 3.6 Distribution of duplicate sequential attacks 3-pattern of malware for a year

3.3.6 Distribution of Activity of Coordinated Attacks

Figure 3.6 shows the distributions of the duplicate sequential attack 3-pattern that were most frequently downloaded by both honeypots, where the *X*-axes indicates the day of the year and the *Y*-axes indicates the download frequency in slots/day. The most common duplicate patterns are *PE_VIRUT.AV* and *PE_BOBAX.AK*. Sequential attack patterns are distributed uniformly during the year. As shown in Fig. 3.6(a), pattern $P_{3.1203}$ has two peaks in February and March 2009 with 10 *slots/day*, whereas pattern $P_{3.857}$ is observed at the maximum rate of 11 slots/day at the end of July 2008. Similarly, Fig. 3.6(b) shows that pattern $P_{3.1203}$ in Honeypot 2 peaks at a similar infection date as the same pattern in Honeypot 1, but with an infection rate of 12 *slots/day*. Pattern $P_{3.857}$ in Honeypot 2 has a peak in September 2008 of nine *slots/day*.

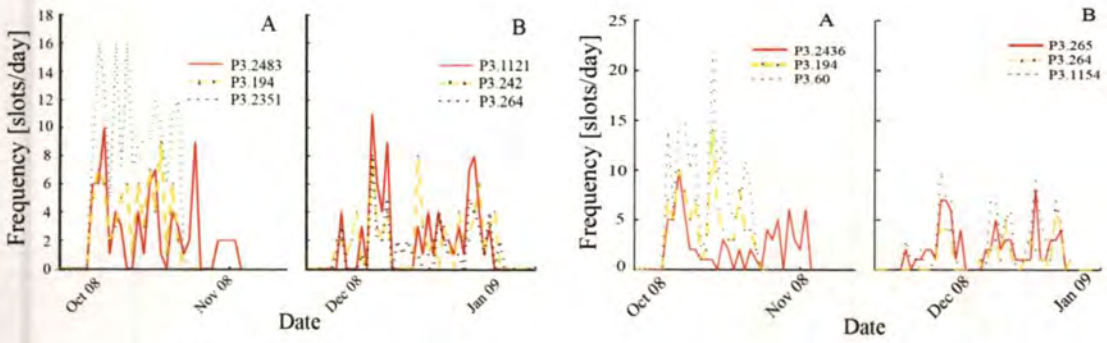
These two patterns in the honeypots are associated with malware that has the ability to disable some services on systems running Windows 2000 and Windows XP such as *Internet Connection Firewall* and *Internet Connection Sharing*. They listen to various ports and connect to an IRC server, and their potential for damage and propagation is rated as medium to high [69]. Regarding the botnet attack, we

conjecture that the distribution diagram shown in Fig. 6 can be considered a distribution of the C&C activity of a botnet system.

3.3.7 Some Classes of Coordinated Attacks

The similarity of either in the malware name or download time exist in the nonduplicate sequential attacks 3-pattern. Group A consist of the same of five malware names that builds all sequential attacks 3-pattern. But there are slightly differences in the sequence of place of malware which compose the sequential attacks 3-pattern. Only pattern $P_{3,194}$ that appears in both honeypots, even though has a bit difference in download frequency by 2 *slots/year*. Similarly, Group *B*, as shown in Table 3.6, includes four identical malware items for both honeypots, but their order is partially reversed. Pattern $P_{3,264}$ infects both of honeypots but differs in download frequency at around 11 *slots/year*.

Figure 3.7 shows the distributions for nonduplicate sequential attack 3-pattern captured during one year and their classification into Groups *A* and *B*, as described above. Both honeypots observed Group *A*, and found that it was downloaded within a 20-day period in October 2008. The maximum infection rate for each honeypot is 16 slots/day and 22 *slots/day*, respectively, as shown in Figs 3.7(a-A) and 3.7(b-A). The activity of botnet attacks in Group *B* looks like a sustained burst (see [34]) throughout 25 days. The activity ran from December 2008 to January 2009, and the maximum infection rate of 11 *slots/day* occurred at Honeypot 1, as indicated by Figs 3.7(a-B) and 3.7(b-B).



(a) Honeypot 1 (b) Honeypot 2

Fig. 3.7 Distribution of non-duplicate sequential attacks 3-pattern of malware for a year

This investigation found great similarities between the two honeypots. There are two common features of nonduplicate sequential attack 3-pattern. Firstly, the pattern attacked intensively for a short period, less than a month in a whole year. Secondly, the number of slots infected is greater than the number of duplicate sequential attack 3-pattern.

In this research, the time intervals for sequential attack 3-pattern downloaded by honeypots are investigated as well. The time interval is defined as the difference in time between the first and last malware item downloaded within a sequential attack 3-pattern. The average (*Avg*) and the standard deviation (*SD*) of time intervals of 3-pattern are shown in Table 3.6. The distribution for each average time interval varies considerably. This may be a result of the dynamic behavior of Internet traffic causing gaps in the time interval for some download events or it may be caused by multiple botnet attacks.

Pattern $P_{3.194}$ in both honeypots has an average time interval of less than 7 minutes and a standard deviation greater than 7 minutes, but the distribution of its time interval, as shown in Fig. 3.8, indicates that the time interval mostly takes only a few values. This means that these patterns were executed at fixed constant time intervals, and it is therefore evidence that the patterns in Group *A* were sent from the same botnet system.

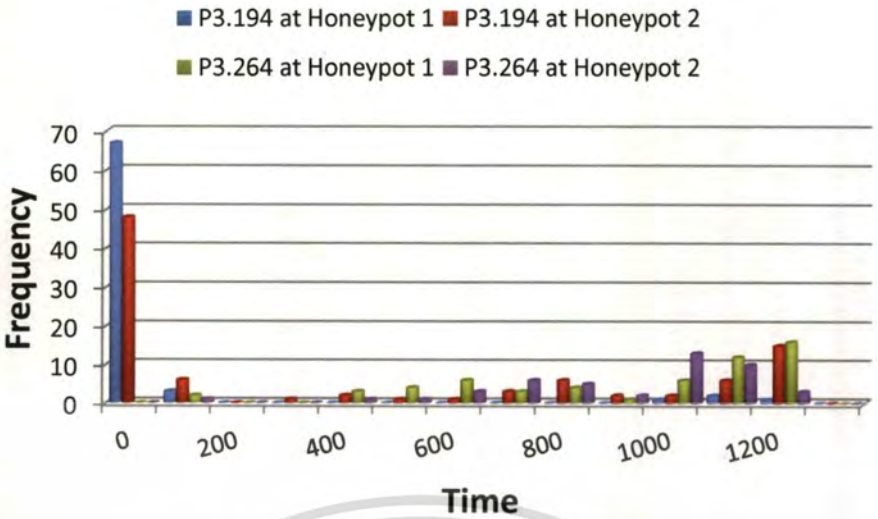


Fig. 3.8 Histrogram of time intervals of the sequential attacks 3-pattern of malware

Conversely, pattern $P_{3.264}$ in both honeypots has average time intervals greater than 14 minutes and standard deviations of less than 6 minutes, but the histogram of its time interval, in Fig. 3.8, indicates that the time interval is randomly spread and widely distributed. Therefore, we claim that these patterns in Group *B* are an outcome of a collision of attacks made by a variety of botnets.

3.3.8 Performance of Mining Sequential Attack Patterns

The mining CCC datasets for discovering the sequential attack 2-pattern and 3-pattern has been ran on a machine with a 2.00 GHz Intel® Core™ 2 Duo T5750 CPU supported by the Ubuntu 10.10 Operating System with GCC version 4.4.5. The *PrefixSpan* algorithm was written in the C++ programming language.

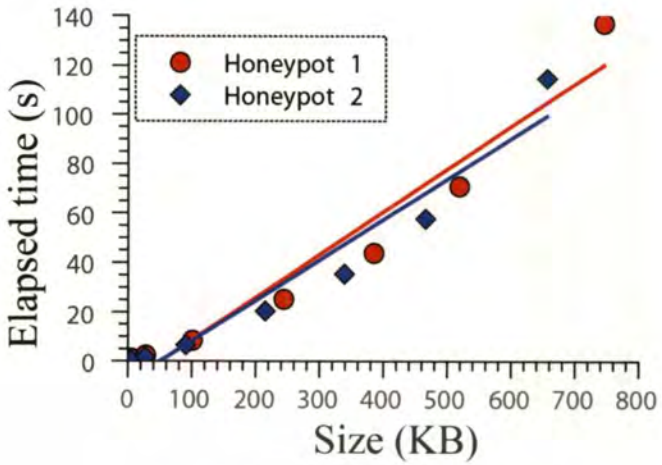


Fig. 3.9 Performance of the *PerfixSpan* method on mining sequential attacks pattern

Figure 3.9 shows the performance of *PrefixSpan* algorithm on mining sequential attack patterns. Performance test has involved pre-processing data with several sizes of files in two honeypots. The pre-processing data are varied from a day, a week, a month, 3 months, 6 months and a year which are represented in the *X*-axes. A general performance test has been taken as elapsed time of computation *s* in the *Y*-axes. The slopes show the performance of *PrefixSpan*, 47.058 bps and 50.000 bps in honeypot #1 and #2, respectively.

3.4 Entropy Analysis of the Sequential Attack Patterns

The mining the CCC dataset of 94 honeypots have generated thousands of sequential attack patterns. The exposure of such a large quantity of valuable information from the discovery of sequential attack patterns raises another challenge. Classifying the sequential patterns in terms of entropy will help us to understand how common some patterns are in attacks on computer networks, and thereby identify significant behaviors in sequential attack patterns.

For each honeypot of 94 in total, the results of mining sequential patterns are sorted by highest to lowest based on the download frequency. Next, pick up the

sequential attack pattern that has the highest frequency of download from each honeypot. By doing so, according to the number of honeypots, it yields the maximum the number of sequential attack pattern are 94. This selection is addressed for both of the duplicate and nonduplicate patterns.

The entropy of the sequential pattern *S* in honeypots is defined by

$$H(S) = -\sum_{i=1}^{I} P(S_i) \log_2\big(P(S_i)\big). \quad (3.1)$$

where $P(S_i)$ is the probability that the sequential pattern *S* attempts to attack the *i*-th honeypot and *I* is the number of honeypots. The probability that each sequential pattern attempts to attack a honeypot is the same. For example, if there are 10 honeypots infected by the sequential pattern *S*, then the probability of the sequential pattern is *P(S1)=P(S2)=P(S3)=...=P(S10)=0.1*. If the maximum number of honeypots *I* is 94, then the entropy score will be in the range $0 \leq H(S) \leq \log_2 94$. The results of the calculation are shown in Table 3.7, where the upper group of rows contains duplicate patterns, and the lower group of rows contains nonduplicate patterns.

The duplicate pattern $P_{3.1203}$ in the upper group and the nonduplicate pattern $P_{3.194}$ in the lower group of Table 3.7 have the highest entropy scores. This indicates that these patterns have attempted to attack the most honeypots. Further examination of their investigation attack distributions reveals that the two patterns have different characteristics. Figure 3.10 (a) shows that pattern $P_{3.1203}$ is distributed uniformly over one year. However, pattern $P_{3.194}$ has a narrow distribution at a specific date and a short duration, as shown in Fig. 3.10 (b). The behavior of pattern $P_{3.1203}$ suggests that malware involving this pattern is commonly used by several botnets. In contrast, pattern $P_{3.194}$ is seen at most honeypots, but it infects at a

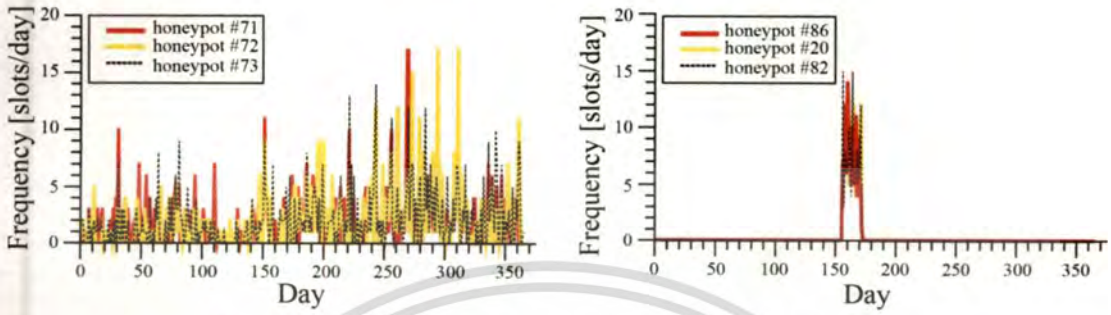
specific date and for a short period. Therefore, it can be deduced that pattern $P_{3.194}$ was sent by a particular botnet for a particular attacking purpose.

Table 3.7 Entropy of the sequential attacks 3-pattern for all honeypots

ID	Pattern Name	Entropy
P3.1203	PE_VIRUT.AV PE_VIRUT.AV PE_VIRUT.AV	6.0875
P3.2425	TSPY_KOLABC.CH TSPY_KOLABC.CH TSPY_KOLABC.CH	5.9307
P3.1590	PE_VIRUT.D-4 PE_VIRUT.D-4 PE_VIRUT.D-4	5.8826
P3.857	PE_BOBAX.AK PE_BOBAX.AK PE_BOBAX.AK	5.8073
P3.1463	PE_VIRUT.D-1 PE_VIRUT.D-1 PE_VIRUT.D-1	5.7814
⋮	⋮	⋮
P3.2796	WORM_RBOT.GDJ WORM_RBOT.GDJ WORM_RBOT.GDJ	2
P3.2528	TSPY_ONLINEG.TKJ TSPY_ONLINEG.TKJ TSPY_ONLINEG.TKJ	1.585
P3.2676	WORM_POEBOT.AKE TSPY_KOLABC.CH TSPY_KOLABC.CH	1
P3.2611	WORM_KOLABC.BQ PE_VIRUT.YE WORM_KOLABC.BQ	0
P3.1924	PE_VIRUT.YC PE_VIRUT.YC PE_VIRUT.YC	0
P3.194	BKDR_RBOT.CZO WORM_HAMWEQ.AP TROJ_QHOST.WT	5.9307
P3.242	BKDR_SCRYPT.ZHB BKDR_SDBOT.BU BKDR_VANBOT.HI	5.7279
P3.2351	TROJ_QHOST.WT WORM_HAMWEQ.AP BKDR_POEBOT.AHP	5.6724
P3.134	BKDR_POEBOT.GN TSPY_KOLABC.CH WORM_SWTYMLAI.CD	5.5849
P3.1368	PE_VIRUT.AV TSPY_KOLABC.CH WORM_SWTYMLAI.CD	5.5546
⋮	⋮	⋮
P3.635	BKDR_VANBOT.FM TROJ_PROXY.WE TROJ_PACK.DT	1
P3.714	BKDR_VANBOT.LE TROJ_BUZUS.ADZ WORM_SPYBOT.ADS	1
P3.1336	PE_VIRUT.AV TROJ_PROXY.ADI TROJ_NOUPDATE.G	0
P3.2659	WORM_POEBOT.AKE BKDR_POEBOT.GN TSPY_KOLABC.CH	0
P3.2641	WORM_PAKES.ABU PE_BOBAX.AK BKDR_VANBOT.LE	0

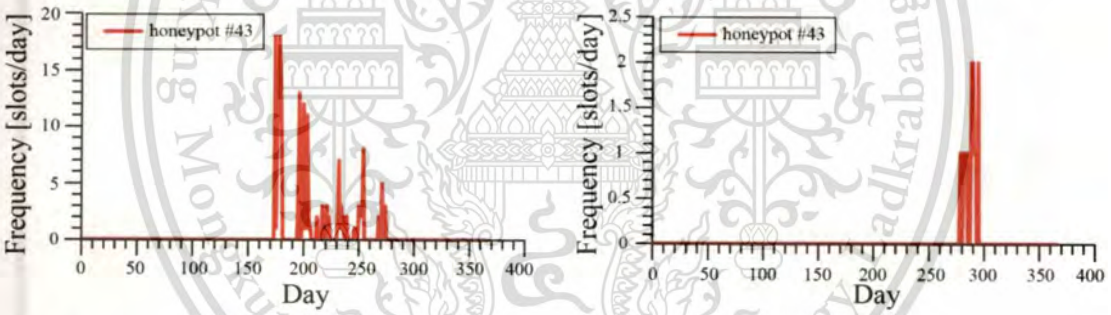
The duplicate pattern $P_{3.1924}$ in the upper group and the nonduplicate pattern $P_{3.2659}$ in the lower group each has a low entropy score, as shown in Table 3.7. The

distribution of attacks for these kinds of patterns is seen for only one honeypot, as shown in Fig. 3.11. They are therefore probably either false detections of a pattern or accidental attacks by inexperienced users.



(a) $P_{3.1203}$ (b) $P_{3.194}$

Fig. 3.10 Distribution of sequential attacks 3-pattern on three different honeypots that have high entropy score for a year: (a) pattern $P_{3.1203}$ and (b) pattern $P_{3.194}$



(a) $P_{3.1924}$ (b) $P_{3.2659}$

Fig. 3.11 Distribution of sequential attacks 3-pattern at one honeypot that has low entropy score for a year: (a) pattern $P_{3.1924}$ and (b) pattern $P_{3.2659}$

3.5 Some Potential Applications

Back to the Chapter 2, Literature Review, some researcher [33-36] have mainly utilized clustering methods for probing botnet attacks. Network flows are classified based on certain criteria such as application communities, source IPs, destination IPs, active time, content length, similar malicious activities and so on. Clustering process doesn't care enough to the sequences of malicious activity, but it

is paid more attention to the similarity of clustering criteria. Conversely, in fact, botnet attacks are established in a sequential manner. Then it becomes difficult to detect a new malicious acvitity or botnet attack use clustering method.

Methods mentioned above are not able to work for sequential attacks because of malwares downloaded by honeypots have specific sequential patterns on various attributes such as malware name (based on hash value), source IP addresses and time interval among malware sequences. An important key point is a *sequential pattern of malware* downloaded by honeypots in the continuous real time.

Analysis results give us great challenges to explore some possible applications. Nowadays a lot of network security applications offer various methods to overcoming the dark side threats of the Internet. A lot of potential applications can be achieved from these results, and three of them will be explained such as a new Intrusion Detection and Prevention System (IDPS); a new botnets firewall which block out botnet sources; and tracking botnets which is possible to identify the sources of malware sent by botnets and estimates the size of botnet.

The corresponding potential applications mentioned previously are briefly described in following:

1) IDPS: The result of the proposed method is considerable to be a new type of the network-based IDPS. Because botnets can generate many kinds of attacks which hard for a specific IDPS technology for overcoming overall attacks. Thus, monitoring and identifying suspicious activity based on the sequential attack patterns of malware downloaded by honeypot offers a wider scope of IDPS. It implies that if we can anticipate botnets, then we can eliminate many kinds of attacks which come from botnets.
2) Botnet Firewall: The classification and behaviors shown in Fig. 3.7 give valuable information as attack alerts. The attacker needs 20 to 25 days (as mentioned in Subsection 3.2.7) to establish an infected computer network as a botnet system. If the first day of infection by an attacker is identified, we can take action to

prevent an ongoing threat. By mining periodically, we can obtain real-time statistics similar to those shown in Fig. 3.7. Therefore, the first day of infection can be identified by monitoring the download frequency, and it could be a start point to block out botnet sources.

3) Tracking Botnet: Due to botnets use systematic attack methods, the sequences of malware downloaded by honeypots have particular forms of coordinated pattern, as shown in Fig. 3.5. Those are valuable information which reveals specific attack for tracking botnets, where malware come from and how big the botnets are. Moreover, the sequential attack patterns of malware can explain botnet strategies how to compromise with victim machines. Source IP addresses belong to the sequential attack patterns of malware are considered as an evidence for network forensics.

Chapter 4

Behaviors of Complex Robotic Networks

4.1 Introduction

The real world network such as social networks, ecological networks, transportation networks, world wide web (WWW) networks, neural networks and so on inspire the elaboration of the complex networks [22, 62, 70]. In the previous chapter, the sequential behavior of malware spreading through the Internet to establish the botnet has been explained. Whereas the chaotic behavior is another one that exist in the complex network as the dynamical system [22, 71]. In [23], Sprott has inspired us with the chaotic behavior of his minimum complex network to extend, realize, and investigate it in the form of complex robotic network. This chapter is sort of extension to extend that of similar to Sprott's framework to the robotic networks. This network will finally be found that it exhibits chaotic behaviors.

During the past decade, chaos has been implemented into a single and multiple chaotic mobile robots by embedding chaotic equations into the robot controller. However, this method has a drawback in fixed chaotic patterns, self-localization and path planning. Alternatively, one also may converts ordinary mobile robots into chaotic ones by implementing a simple interaction among them. In parallel with the one of objectives of this research is to develop a new model of a chaotic mobile robot network without embedding a chaotic system equation into the mobile robot controller.

To address this goal, a special complex network in which the dynamics of the global behaviors of the system emerge from a strongly interconnect among the agents can be highly considered. In fact, networks of dynamical systems have been used to model a wide range of complex systems [22]. The number of nodes in the

network as small as four can behave chaotically with proper interaction non-linearly [23, 63]. Adopting this fact, four ordinary mobile robots are employed to face this challenge.

This chapter will explain the simulation of a new method of a non-embedded complex robotic network. Started from a concept that combines the inverse square law of a distance to modify the minimum complex network that presented in [23]. Then it continues to the section that discusses the system realization into mathematical expression. Afterward, a simple method to avoid wall congestion and collision will be presented. The next section will simulate the system being investigated. Then it's followed by the discussion of evaluation criteria to measure the quality of interaction among robots. The last two sections will analyze the behavior of complex robotic network in two cases of open-workspace and closed-workspace, respectively. Finally, performance evaluation on comparison with the embedded counterparts will be discussed.

4.2 Conceptual Model of the Non-Embedded Complex Robotic Network

A two-wheeled mobile robot is assumed as an agent in the colony. Each mobile robot herein is identical design both in functional and physical specification with a $0.08m$ diameter in circular shape. To create a complex robotic network, the proposed scheme of the minimum complex network appeared in [23] is adopted.

In [23], Sprott validates the existence of the minimum complex network of an artificial neural network. In our work, however, the network is realized by four mobile robots interacting among themselves and their environment. Unlike the interaction in [23], this model is based sound communication. The technique is adopted from insect's communication using sound as input command to other autonomous mobile robots [72]. In this computer simulation, this feature of insect-like called audible-inaudible scenario is modified from the previous work [64]. Adopting this scene, when

every robot receives the sound emitted by others, it then moves in accordance with the received sound energy. The higher intensity the sound is, the higher the response will be. But, this research is actually inspired by the law behind the sound energy, that sound intensity emitted from transmitter to the receiver obeys the inverse square law of distance. Thus, it is not limited for using only sound media to communicate. Illustration of the scheme is shown in Fig. 4.1.

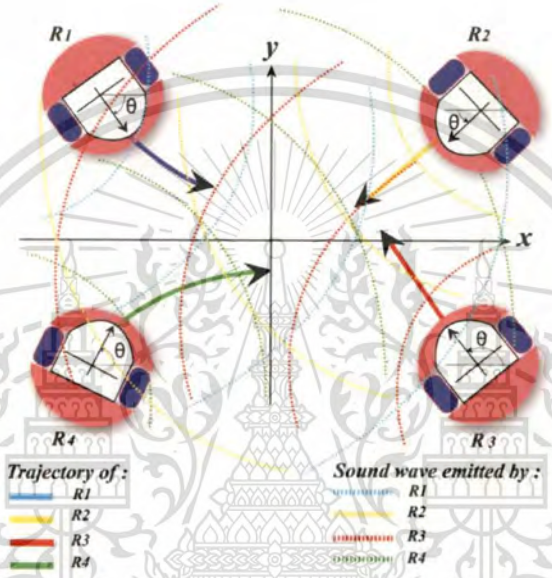


Fig. 4.1 Four mobile robots forming a dynamic network based on their interaction

4.3 System Realization of Non-Embedded Complex Robotic Network

There are four robots compose a network by interacting with the others. Each robot is able to hear the sound emitted from its peers. In other words, it is able to receive sound energy I. Since the interaction among the network elements in [23] is not too fitted the robotic paradigm, the inverse square law is introduced instead. The proposed interaction is fitted most mechatronic system. Both opto, audio, and mechanical subsystems are obeyed this law. Physically I is inversely proportional to the square of distance between the robots. Mathematically the sound energy I received by the robot is described by

$$I \propto \frac{1}{r^2}, \tag{4.1}$$

where r represents the distance between the robots.

The interaction mechanism among robots can be described as follows. Each robot first receives sound energy that is emitted by the other nearby robots, and then sums up the sound energy received to carry out the interaction among themselves. Afterward, the total energy will be used for driving its wheels. Recall that the system evolved in time can be represented as the *Ordinary Differential Equation* (ODE). The state equation of this communication scenario can be described mathematically as

$$\dot{s}_i = -w_i s_i + g \sum_{\substack{j=1 \\ i \neq j}}^{N} I_{ij}, \tag{4.2}$$

where N is the number of robots. The term w_i is a coefficient or a coupling constant, which represents the intentional interaction of the robot s_i, and g is constant coefficient with default is 1. The i^{th} robot receives sound energy I, which is emitted by the j^{th} robot. Hence, the interacting intensity is represented as I_{ij}. The robot does not receive sound energy I emitted by itself, which is indicated mathematically as $i \neq j$. Assume that a notch filter is implemented to cancel its own frequency and each robot produces a unique frequency. Strong interaction among robots is equivalent to decreasing a value of w_i. To ensure a solution, we define $w_i > 0$. For convenience, this simulation is given $w_i = w$ for all i. The coupling coefficient w can be viewed as a gain associated with the dynamically interacting variable s_i. We now substitute I_{ij} in (4.2) with $\frac{1}{r^2}$ from (4.1), and assign a unit constant. Thus, we obtain the interaction state equation of the robot as

$$\dot{s}_i = -ws_i + g\sum_{\substack{i\neq j \\ j=1}}^{N} \frac{1}{r_{ij}^2}. \tag{4.3}$$

Realizing physical structure, Fig. 2.5 displays a two-wheeled mobile robot that in use. Movement of the robot along the navigation path follows the Eq. (2.20), where v [m/s] is the linear velocity of the robot, and ω [rad/s] represents the angular velocity. The position of the mobile robot i^{th} in the *Euclidean* space $\mathbb{R}^2$ is determined by x_i and y_i [m].

In order to generate chaotic movement of the robots, we combine (2.20) and (4.3). The solution of the interaction model in (4.3) will be an input for the kinematic model in (2.20). Then the system model of the robot in the network is

$$\begin{cases} \dot{s}_i = -ws_i + g\sum_{\substack{i\neq j \\ j=1}}^{N} \frac{1}{r_{ij}^2} \\ \dot{x}_i = v\cos(s_i) \\ \dot{y}_i = v\sin(s_i) \end{cases}, \tag{4.4}$$

where (x_i , y_i) is a pair of coordinates on the *Cartesian* space that represent the i^{th} robot's position at the point x_i and y_i, respectively. The distance between two mobile robots i^{th} and j^{th} in the plane $\mathbb{R}^2$ is the *Euclidean* distance r_{ij} , which is defined as

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{4.5}$$

This research is conducted by setting N equal to 4 as the number of robots in the minimum non-embedded complex robotic network. If we substitute r_{ij} in (4.4) with (4.5), then we get the state equation of the system for the minimum non-embedded complex robotic network which is stated as

$$\begin{cases} \dot{s}_i = -ws_i + g\sum_{\substack{j=1 \\ i\neq j}}^{4} \frac{1}{(x_i-x_j)^2+(y_i-y_j)^2} \\ \dot{x}_i = v\cos(s_i) \\ \dot{y}_i = v\sin(s_i) \end{cases}, \tag{4.6}$$

where v [m/s] is the linear velocity of the robot.

4.4 Simple Avoidance Strategy for Wall Congestion and Collisions

A strategy for avoiding congestions and collisions is important to make robots move naturally. In case of the mobile robot is designed to travelling inside the boundary of a predefined workspace, and thus a non-zero probability of the robot collision with obstacles or another robot cannot be avoided. To cope with these constraints of congestion and collision avoidance, a mirror mapping technique is implemented [17, 18, 20, 21]. The technique suggests that if the robot approaches to the wall or an obstacle at an incident angle α, then the robot must bouncing back at the same reflection angle α.

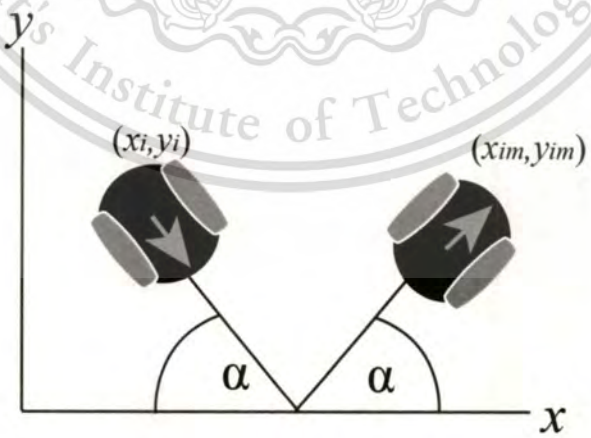


Fig. 4.2 Mirror mapping technique reflects the robot with the angle equal to the incident α

The algorithm is a simple as follows. When the robot hits the wall, the robot will be reflected back inside the workspace. Employing the law of refelction, the reflection angle is the same as the incident angle. Figure 4.2 illustrates the mirror mapping technique. According to the figure, the mobile robot hits the wall with an incident angle **α**., and then maps to the new position with the angle is exactly equal to the incident one.

Suppose that x_{left}, x_{right}, y_{max} and y_{min} are the edges of the workspace for left, right, upper and lower on *Cartesian* coordinate, respectively. Initially, the robot position is (x_i,y_i), if the next position (x_{i+1},y_{i+1}) will cross the border, then the mirror mapped point when the robot crosses one of the border is (x_{im}, y_{im}), it can be defined as follow;

$$\begin{aligned} &left: x_{im} = 2x_{left} - x_{(i+1)}, && y_{im} = y_i; \\ &right: x_{im} = 2x_{right} - x_{(i+1)}, && y_{im} = y_i; \\ &upper: x_{im} = x_i, && y_{im} = 2y_{max} - y_{(i+1)}; \\ &lower: x_{im} = x_{im}, && y_{im} = 2y_{min} - y_{(i+1)}. \end{aligned} \tag{4.7}$$

In a situation that the robot collides to each other, a simple strategy is implemented. Assume that each robot has a circle shape with radius 0.04m and a safe area with the radius 0.08*m* from its center of body. When two or more robots move coincide into this area, then robots stop in the current position to avoid congestion. Next, each robot waits for a new free-congestion path generated by the evolution of the dynamic system. This idea is very simple, stops in the current position and generates a new path. This simplicity is suitable because the chaotic behavior of the robot does not pay attention to a specific path or destination.

4.5 Numerical Calculation of the Non-Embedded Complex Robotic Network

Dynamical system is something that deals with change, in which a system such non-embedding chaotic robot network in Eq. (4.6) evolves in time. To enumerate efficiently the solution of such system, a method of numerical calculation should be selected. One of the famous and broadly implemented is the fourth-order *Runge-Kutta* method.

The fourth-order *Runge-Kutta* integrator is widely used in the numerical calculation for a solution of the ODE of dynamical system including of the embedded chaotic mobile robot. Generally, *Runge-Kutta* integrator traverses a solution along an interval by mixing the information from several *Euler*-style steps that each involving one evaluation of the right-hand f's, and then using the information obtained to find a *Taylor* series expansion up to some higher order. This method is easy to implement into a computation, and fastest method for evaluating the cheap f_i [61]. Fourth-order *Runge-Kutta* method defines the following formula;

$$\begin{aligned} k_1 &= hf(t_n, \rho_n) \\ k_2 &= hf\left(t_n + \tfrac{1}{2}h, \rho_n + \tfrac{1}{2}k_1\right) \\ k_3 &= hf\left(t_n + \tfrac{1}{2}h, \rho_n + \tfrac{1}{2}k_2\right) \\ k_4 &= hf\left(t_n + \tfrac{1}{2}h, \rho_n + \tfrac{1}{2}k_3\right) \\ \rho_{n+1} &= \rho_n + \tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4 \end{aligned} \qquad (4.8)$$

In each step the derivative is executed four times. Firstly, f_i is evaluated at the initial points. The second and third are executed at midpoints. At last is executed at a trial endpoint. From these derivatives the final point is defined.

Equation (4.8) is an evolution machine for generating trajectories of robots. In order to generate trajectories, Eq. (4.8) evaluates Eq. (4.6) for each step size h propagates in time. Thus, the trajectory of robot i^{th} is achieved from last two parts of Eq. (4.6), that is (x_i, y_i). In case of the robot forbids traveling beyond a predefined

workspace, the last fifth part of Eq. (4.8) $\rho_{n+1} = \rho_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4$ requires to be improved. By doing so, Eq. (4.8) can be defined specifically for this purpose as

$$\begin{aligned}
&k_1 = hf(t_n, \rho_n);\\
&k_2 = hf\left(t_n + \tfrac{1}{2}h, \rho_n + \tfrac{1}{2}k_1\right);\\
&k_3 = hf\left(t_n + \tfrac{1}{2}h, \rho_n + \tfrac{1}{2}k_2\right);\\
&k_4 = hf\left(t_n + \tfrac{1}{2}h, \rho_n + \tfrac{1}{2}k_3\right);\\
&\rho_{n+1} = \rho_n + \tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4, \quad for\ \rho_n = s_i;\\
&\rho_{n+1} = \rho_n + c_n\left(\tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4\right), for\ \{\rho_n = x_i | \rho_n = y_i\}
\end{aligned} \quad . \qquad (4.9)$$

The coefficient c_n is toggle between 1 and -1 that can be described as

$$c_n = \begin{cases} 1 & \{(x_i < x_{left}) | (y_i < y_{lower})\} \\ -1 & \{(x_i > x_{right}) | (y_i > y_{upper})\} \end{cases}. \qquad (4.10)$$

In the other words, the coefficient c_n plays the role of the motion direction of robots when they collide to the wall/boundary. Consequently, when the robot i^{th} moves beyond the left or lower than the boundary, coefficient c_n can make the robot i^{th} return back into the workspace.

The solution of the numerical calculation above will generate trajectories of four non-embedded chaotic mobile robots. The trajectories come up from strands of the solution of x_i and y_i (the last two part of Eq. 4.6) in the *Cartesian* coordinate. Behavior of robots creates chaotic trajectories on the workspace that can be observed in the computer simulation.

4.6 Computer Simulation

Computer simulation aims to visualize the behavior of the non-embedded complex robotic network in the form of trajectory. This dynamic visualization of

trajectories is able to reveal the state space of the system observed visually. Adopting this benefit, four non-embedded chaotic mobile robots network are simulated into 2-D of *Cartesian* coordinate. Solutions of Eq. (4.6) are evaluated by Eq. (4.9) for some duration of times, and are plotted.

This work provides the Graphical User Interface (GUI) to make the simulation friendlier to be operated. The GUI has been developed by using C++ programming language. It is able to record trajectories of four robots into the text files for further investigation.

This research defines some constraints of the physical body of robot and its environment to satisfy the simulation purpose. The robot is assumed to have a circle shape with radius $0.04m$. The environment is set as a flat workspace. To simplify the simulation, the velocity for each robot is set to a constant linear at $0.05m/s$. Since the chaotic system sensitively depends on the small difference in the initial condition and thus six different initial conditions are defined based on the orientation of robots in the *Cartesian* coordinate. With this in mind, a term scenario (P) for a set of each different initial condition is used.

Table 4.1 Initial conditions of robots' orientation angles of six scenarios

Scenario #	Initial Conditions	Figure
P_1	$s_1 = 5.4978, s_2 = 3.7525, s_3 = 2.3562, s_4 = 0.7854$	4.3(a)
P_2	$s_1 = 2.3562, s_2 = 0.7854, s_3 = 5.4978, s_4 = 3.7525$	4.3(b)
P_3	$s_1 = 2.3562, s_2 = 3.7525, s_3 = 2.3562, s_4 = 0.7854$	4.3(c)
P_4	$s_1 = 5.4978, s_2 = 0.7854, s_3 = 2.3562, s_4 = 0.7854$	4.3(d)
P_5	$s_1 = 5.4978, s_2 = 0.7854, s_3 = 2.3562, s_4 = 3.7525$	4.3(e)
P_6	$s_1 = 2.3562, s_2 = 3.7525, s_3 = 0.7854, s_4 = 3.7525$	4.3(f)

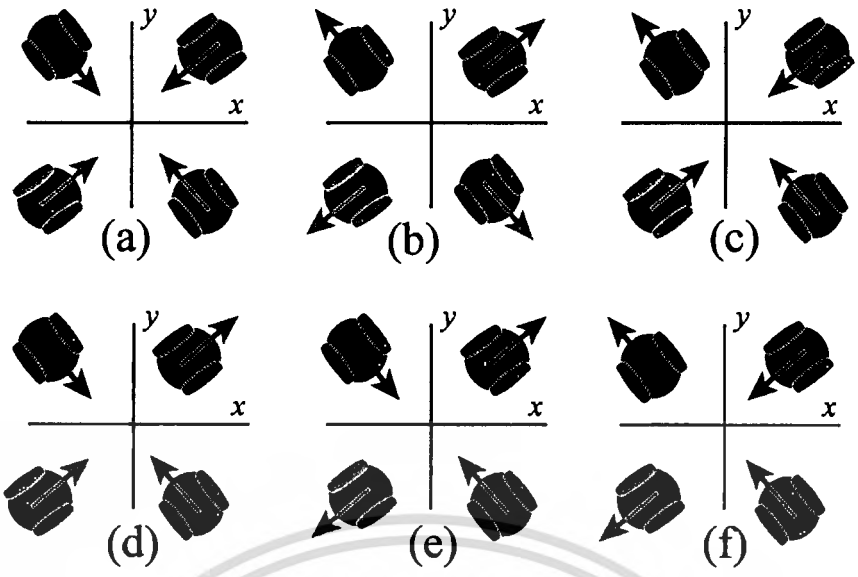


Fig. 4.3 Initial conditions of robots' orientation angles of six scenarios

The initial location (starting point) on the *Cartesian* coordinate for each robot (x_i, y_i) in (4.6) are set to the same for all scenarios, and these are given as follow: $x_1 = -0.5$, $y_1 = 0.5$, $x_2 = 0.5$, $y_2 = 0.5$, $x_3 = 0.5$, $y_3 = -0.5$, $x_4 = -0.5$, and $y_4 = -0.5$. Simultaneously, initial conditions of the angle orientation for each robot is described in Table 4.1 and illustrated in Fig. 4.3.

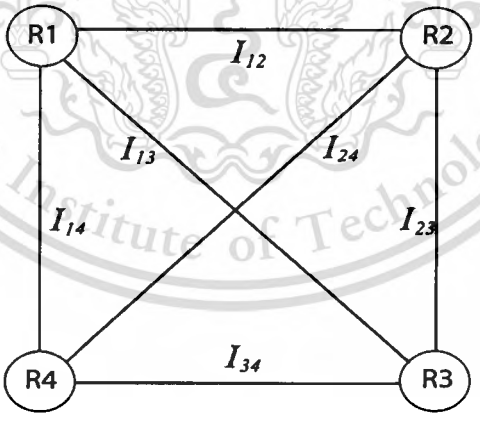


Fig. 4.4 Complete graph represents the interaction of robots in the network, where R_1-R_4 are 1th-4th robots, respectively. Weight of the link between Robots *i* and *j* is marked as l_{ij}

Robotic network interact each other via links that assumed to be established by transmitting and receiving energy among them. Consequently, the quality of the link becomes important to preserve robots interaction in the network. This research evaluates the interaction quality based on the inverse square law of distances among robots that will be explained in the next section.

4.7 Evaluation Criteria: the Interaction Quality and Area Coverage

4.7.1 The Interaction Quality

According to the inverse square law, the weight of the link between two robots decays exponentially due to robots move apart each other. This yields the proportional effect toward the interaction among robots in the network. The *weight* here is defined as a distance between two robots in the workspace. Figure 4.4 shows a complete graph represents the interaction of robots in the network. Our conjecture is the interaction quality has proportional relationship to the chaotic behavior. Here we introduce a method to quantify the association of the interaction quality of robots for the minimum complex robotic network emerged chaotic behaviors.

We now define I_{avg} as the average of the interaction quality among robots at a certain time. In a Graph point of view, maximum number of possible links or edges in the complete graph is $\frac{N(N-1)}{2}$, where N is the number of robots or nodes. Then, we can obtain I_{avg}through the summation of all weight and take the average using a formula written below

$$I_{avg} = \frac{1}{N(N-1)}\left(\sum_{\substack{i=1 \\ i\neq j}}^{N}\sum_{j=1}^{N}\frac{1}{r_{ij}^2}\right), \tag{4.11}$$

where r_{ij} is the distance between robot i^{th} and j^{th} at the certain time.

In order to obtain the average of the total interaction quality among robots during a slot of running time, we accumulate the I_{avg} and take the average relative to the number of steps. The average of total interaction quality among robots during a slot of running time is defined as ξ_{avg}. For this purpose, ξ_{avg} can be expressed as follow

$$\xi_{avg} = \frac{1}{K}\sum_{n=1}^{K}\left(\frac{1}{N(N-1)}\left(\sum_{\substack{i=1 \\ i\neq j}}^{N}\sum_{j=1}^{N}\frac{1}{r_{ij}^{2}}\right)\right), \tag{4.12}$$

where *K* is the number of steps in the slot of time.

This research brings two cases of evaluation for the robot network behaviors. First case is robots' behaviors in the open-workspace that represents a low effect of an environment. Second case is robots' behaviors in the closed-workspace that gives a picture of the environment makes a contribution for robots' behaviors. The next two sections will sequentially discuss both cases open-workspace and closed-workspace, respectively.

4.7.2 The Area Coverage

Many researchers have proposed to implement the chaotic mobile robots toward the home appliances such as lawn mowers or floor-cleaners and industrial applications [14, 18]. Such applications acquire area coverage property. We now evaluate the trajectory performance of the mobile robots for effectiveness in this sense. The area coverage C_A is a ratio of the total coverage of an area A_c, in which the mobile robot's trajectory passes through to the total reachable area A_T, as described below

$$C_A = \frac{A_C}{A_T}. \tag{4.13}$$

Remark that the index C_A defined herein is measured numerically for the areas coverage not via frequency of visiting routes. In other words, it is a capacity measurement, not the informative one. This is because we are now interesting in areas coverage. This formula will be used to evaluate the performance on non-embedded chaotic robot networks. The last section of this chapter will discuss the implementation of this formula in (4.13).

4.8 Analysis Behaviors of the Non-Embedded Complex Robotic Network on Open-Workspace

Open-workspace simulates a condition of robots interact each other in a free flat workspace. The free flat workspace assumes the square area is unlimited for robots exploration. Initial conditions and positions of robots follow the configuration that mentioned in Chapter 4.6. The running time of robots is defined as 83*min*, and it is sliced into three sequential time slots for evaluating the interaction quality. Three slots of time are 0-16*min*, 17-41*min*, and 42-83*min*. For each slot of time the trajectory of 1^{th} robot on scenarios P_1-P_6 are captured and plotted as shown in Fig. 4.5.

Figure 4.5 shows trajectories, which are observed within the first 16*min*, seem to behave chaotically around the neighborhoods of the initial conditions. For the second slot time from 17^{th} *min* to 41^{th}*min* robot tends to move away from the group recognized as weak interaction. The last slot of time 42^{th}*min* through 83^{th}*min* shows the movement of robots totally lost their interaction and move along steady positions and keep moving obey the first law of *Newton*.

Let us look more into the cause of this phenomena quantitatively. Figure 4.6 shows the average of total interaction quality among robots spanning along the three

slots of time. The ξ_{avg} continues decreasingly as the time goes on. This represents the mobile robots freely move farther away from each other in the boundless area. As the robots evolve in time, the distances among them increase to make the average of total interaction quality decreases asymptotically. Chaotic movement of robots cannot be preserved if the average of the total interaction quality among the mobile agent robots ξ_{avg} decrease below 0.5 as shown in the Fig. 4.6.

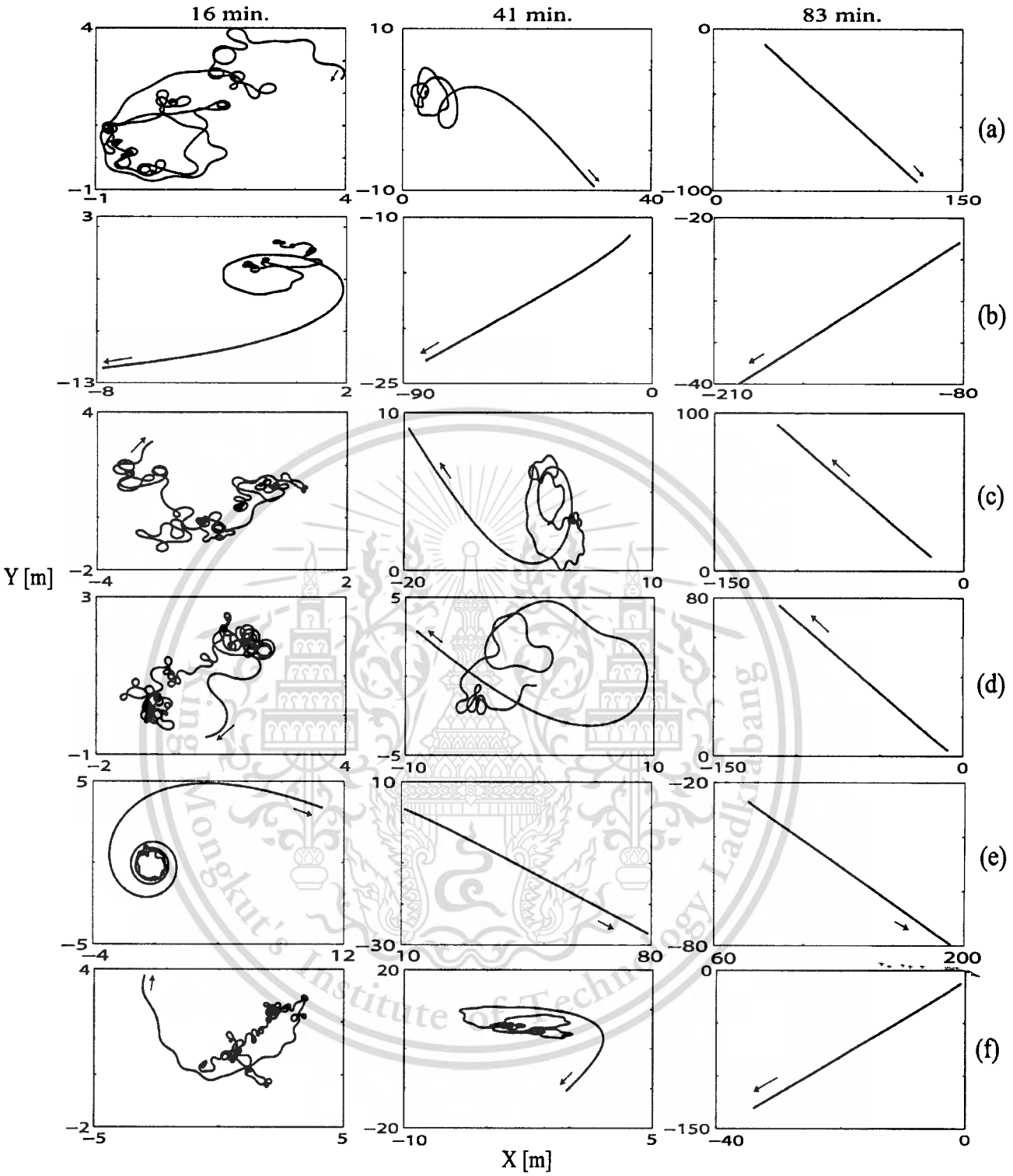


Fig. 4.5 Trajectories of robots have been captured within three different slots of time on six scenarios: (a)-(f) represent trajectories of scenario P_1-P_6, respectively

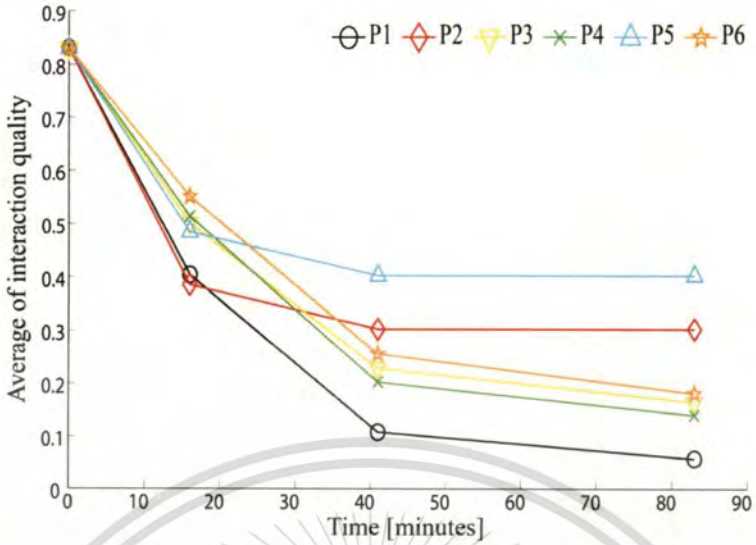
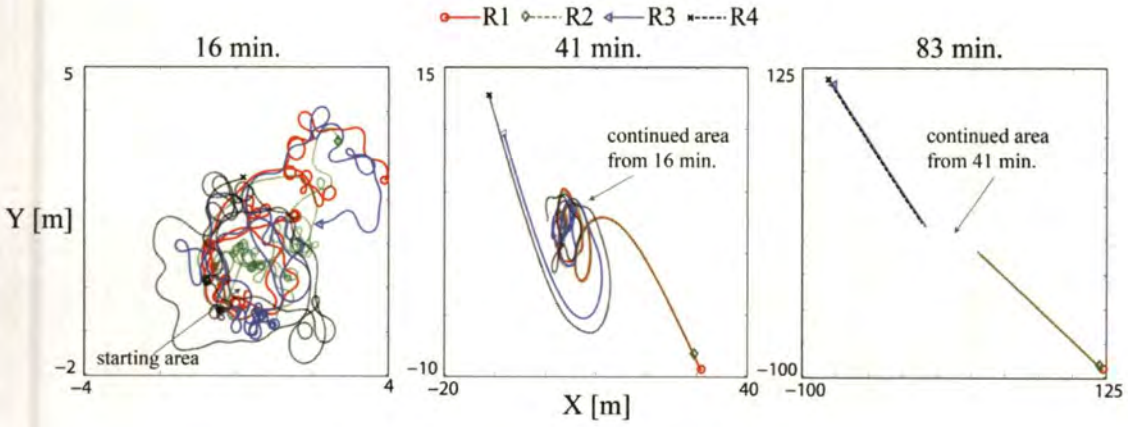
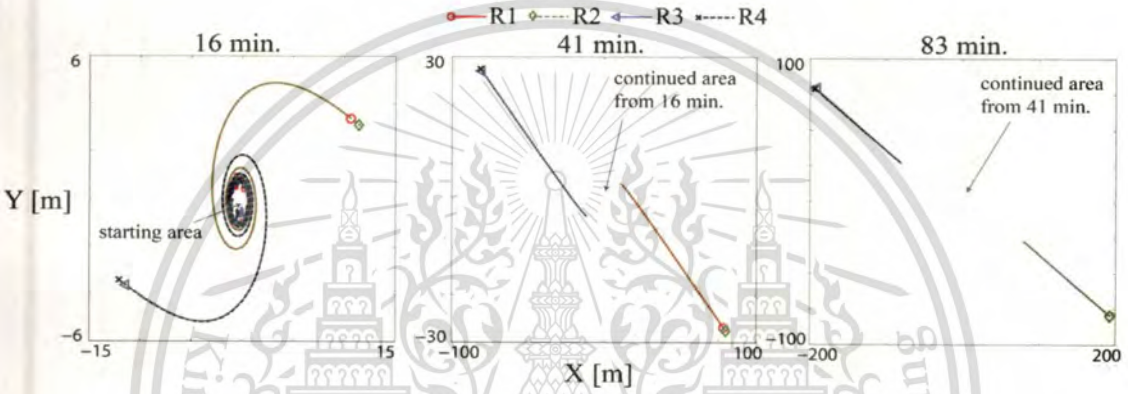


Fig. 4.6 The average of the total interaction quality of robots (ξ_{avg}) travel in open-workspace in six scenarios for 83 minutes

If we observe both Figs. 4.5 and 4.6 in detail, there is an interesting case. The ξ_{avg} of P_1 is less than P_5, although qualitatively P_1 more durable in chaotic behaviors than P_5, as shown in Fig. 4.5(a) and (e). The ξ_{avg} of P_5 decreases slower than P_1, it is caused by robots in the network spreads into two groups move in the opposite direction within first 16^{th} *min*. Meanwhile, each group consisting of two robots move in the same direction and very close. If each robot moves with constant velocity and their distance in each group is very close, it makes the ξ_{avg} of P_5 to be greater than ξ_{avg} of P_1 that moves chaotically with the distance among them dynamically varied. Figure 4.7 shows the dynamical behaviors of four robots in the network both P_1 and P_5. Both of P_1 and P_5 have similar behavior.



(a) trajectories of four robots in P_1



(b) Trajectories of four robots in P_5

Fig. 4.7 Trajectories of robots have been captured within three different time frames in P_1 and P_5

The chaotic behavior is a warranty of a system has property of chaos. The system that sensitively depends on very small difference in the initial condition will bring a nearby trajectory diverge exponentially fast in time. The system that holds this property can be proved by the existence of the largest *Lyapunov* exponent (LLE) is positive and greater than zero [73], and LLE is acceptable to be a signature of the existence of the chaotic behavior [23, 63]. With this in mind, we calculate the LLEs of the system in (4.6) for all P_1 to P_6 run on the open-workspace by using *Wolf* algorithm [58] having 5×10^6 iterations with fixed 0.05 step size of *Runge-Kutta* integrator.

Table 4.2 presents the results of LLEs for six different scenarios P_1-P_6 of the system in (4.6) for the open-workspace. The LLEs are very small and can be considered as zero. These can be evident that there is no chaotic behavior of the system when it runs on the open-workspace.

Table 4.2 The largest Lyapunov exponents of the non-embedded complex robotic network for all scenarios P_1-P_6 on the open and closed workspace

Scenario #	Open-Workspace bit/sec[ii]	Closed-Workspace bit/sec
P_1	≈ 0.0001525	≈ 0.2122825
P_2	≈ 0.0005684	≈ 0.2036364
P_3	≈ 0.0005373	≈ 0.2075451
P_4	≈ 0.0003935	≈ 0.2306058
P_5	≈ 0.0005300	≈ 0.2093000
P_6	≈ 0.0007021	≈ 0.2198831

4.9 Analysis Behaviors of the Non-Embedded Complex Robotic Network on Closed-Workspace

Closed-workspace is another case that represents a situation of robots interact each other and its environment in the flat workspace. The environment is defined as a $2\times2m^2$ workspace. Initial conditions and positions are exactly the same with the previous case (open-workspace) as well as the running time and the number of slots time to be investigated. But, the significant difference in the predetermined

[ii] bits/sec refers to the original paper of Wolf, [58] A. Wolf, J. Swift, H. Swinney, and J. Vastano, "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena,* vol. 16, pp. 285-317, 1985.

area defines a limitation for robots travelling in the workspace. When the robot is travelling beyond the border, it will be forced to return back into the workspace by the mirror mapping technique.

The behavior of robot network changes significantly in a closed-workspace. The chaotic behavior emerges spontaneously from the interaction among robots. Figure 4.8 displays trajectory of Robot 1 on P_5 that shows the chaotic behavior preserved for each slot of time. According to the interaction quality ξ_{avg} in (4.12) and deals with visual evidence shown in Fig. 4.8, and thus we confirm that the important key factor is strong interaction among robots. This phenomena have been strengthened by the average of the total interaction quality among robots, ξ_{avg}, greater than 1, as shown in Fig. 4.9.

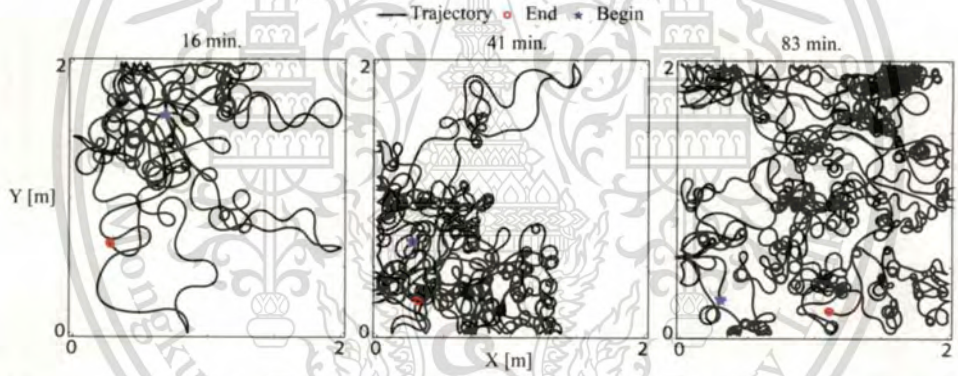


Fig. 4.8 The trajectory of Robot 1 in the scenario P_5 for three different time frames

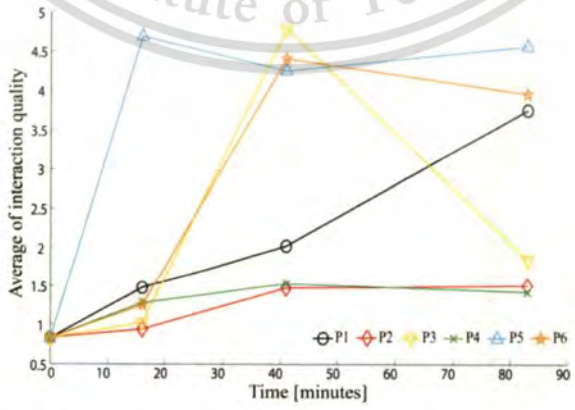


Fig. 4.9 The average of the total interaction quality of robots (ξ_{avg}) travel on the closed-workspace for 83 minutes

As the property of chaos can be probed with the LLE, we calculate the LLEs of the system in (4.6) for all P_1 to P_6 by using *Wolf* algorithm [58] having 5×10^6 iterations with fixed 0.05 step size of *Runge-Kutta* integrator. The LLEs are presented in Table 4.2, and all values of the LLEs for all scenarios are positive. It can be considered as an evidence that the system in (4.6) holds the property of chaos.

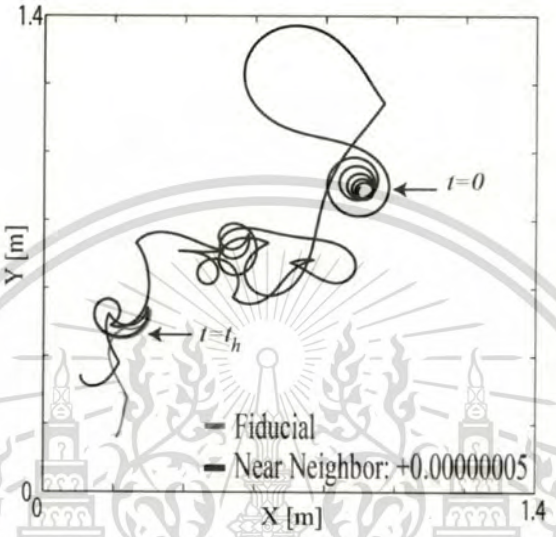


Fig. 4.10 Very small difference in the initial condition makes two trajectories diverge in the future

In the other words, the property of chaos system that is sensitively dependent on very small difference in the initial condition can be seen visually. It can be seen by comparing both of nearby and fiducial time series data, or in this case is both of nearby and fiducial robots trajectories in the workspace. Strogatz [44] reported that a fiducial trajectory very close near neighbor trajectory will rapidly diverge from each other. Figure 4.10 shows a visualization of this phenomenon, where the nearby trajectory starts at time $t=0$ around $+5\times10^{-8}$ can only predict the fiducial trajectory until $t=t_h$, and the rest of the trajectories are completely different. In the other words, beyond the t_h, the predictions are impossible. This visualization supports the existence of chaos in (4.6).

4.10 Performance Evaluation of the Non-Embedded Complex Robotic Network

The motivation of this evaluation is that this type of robots has non-repeatable trajectory providing excellent area coverage [14, 21, 40]. The area coverage property makes the chaotic robots suitable for searching and patrolling as studied in [18, 19]. With this in mind, four identical two-wheeled mobile robots are designed to explore an unknown terrain in a square workspace.

This computer simulation evaluates the performance to validate the effectiveness of the design in a simulation. Six different scenarios on a 2×2 m^2 workspace as mentioned before are applied, in which initial conditions and positions of robots are described in Table 4.1 and illustrated in Fig. 4.3. The proposed paradigm is compared with two embedded chaotic mobile robots on well-known Arnold's and Lorenz's equations.

4.10.1 Arnold's and Lorenz's Equations

To compare the performance, embedded chaotic mobile robots are realized. In doing so, we employ the chaotic equations of Lorenz and Arnold. Note that Lorenz equation represents an attractor-type. And Arnold equation represents a non-attractor one. The technique and parameters for chaotic equations for the mobile robots are similar to those in [18, 21, 37]. A short form of the state equation of the mobile robot which employs the Lorenz's equation is

$$\begin{bmatrix} \dot{L}_1 \\ \dot{L}_2 \\ \dot{L}_3 \\ \dot{x}_L \\ \dot{y}_L \end{bmatrix} = \begin{bmatrix} -\sigma L_1 + \sigma L_2 \\ -L_1 L_3 + \rho L_1 - L_2 \\ L_1 L_2 - \beta L_3 \\ v \cos L_3 \\ v \sin L_1 \end{bmatrix}, \tag{4.14}$$

where L_i are the state variables regarding to Lorenz attractor and σ, ρ, and β are coefficients or constants. The initial condition and coefficient values are presented in Table 4.3.

For Arnold's equation, we have

$$\begin{bmatrix} \dot{Z}_1 \\ \dot{Z}_2 \\ \dot{Z}_3 \\ \dot{x}_Z \\ \dot{y}_Z \end{bmatrix} = \begin{bmatrix} A\sin Z_3 + C\cos Z_2 \\ B\sin Z_1 + A\cos Z_3 \\ C\sin Z_2 + B\cos Z_1 \\ v\cos Z_3 \\ v\sin Z_3 \end{bmatrix}, \tag{4.15}$$

where Z_i is a state variable regarding to Arnold equation and A, B, and C are constants. The initial condition and constants are presented in Table 4.3.

Table 4.3 Initial conditions of Lorenz's and Arnold's systems

System	Initial Conditions
Lorenz	$L_1 = 10, L_2 = 11, L_3 = 10, \sigma = 10, \beta = 8/3, \rho$=28
Arnold	$Z_1 = 2, Z_2 = 2, Z_3 = 0, A = 1, B = 0.5, C = 0$

4.10.2 On Comparison on the Non- and Embedded Chaotic Mobile Robot: Area Coverage

To investigate the area coverage, the index calculation C_A in (4.13) is computed and is snapped shot at the 16th, 41th, and 83rd minutes. We set the robot's speed at 0.05*m/s*. To assign resolution of the workspace, we assign by utilizing the physical diameter of the mobile robot, which is 0.08 m, and the total workspace, which is 4 m^2. The total area A_T is normalized by mapping into 25×25 grid cells or 625 grid cells in total. This is the resolution of the unit cell of the workspace under inspection for Figs. 4.11-4.13

According to the simulation results based on six scenarios, the lowest area coverage is achieved by Robot 1 in Scenario P_2 with C_A=60.48% and the highest area coverage is reached by Robot 3 robot in Scenario P_5 with C_A=97.76%. For the embedded chaotic types, the highest areas coverage C_A generated by Lorenz's and Arnold's equations are 93.44% where the initial point stating on quadrant Q2 and 96.48% where the initial point staring on quadrant Q3, respectively. Figures 4.11 and 4.12 are shown the results, respectively. Note that the colorbar on the right side of the time diagram of figures show the frequency of events occurred when the mobile robot passing through the grids. The total run time is 83 minutes. In the comparison, the evaluation is simple. We just use the paradigm of the best against the best. Having compared Figures 4.11-4.13, it is evident that the non-chaotic type outperforms the embedded ones.

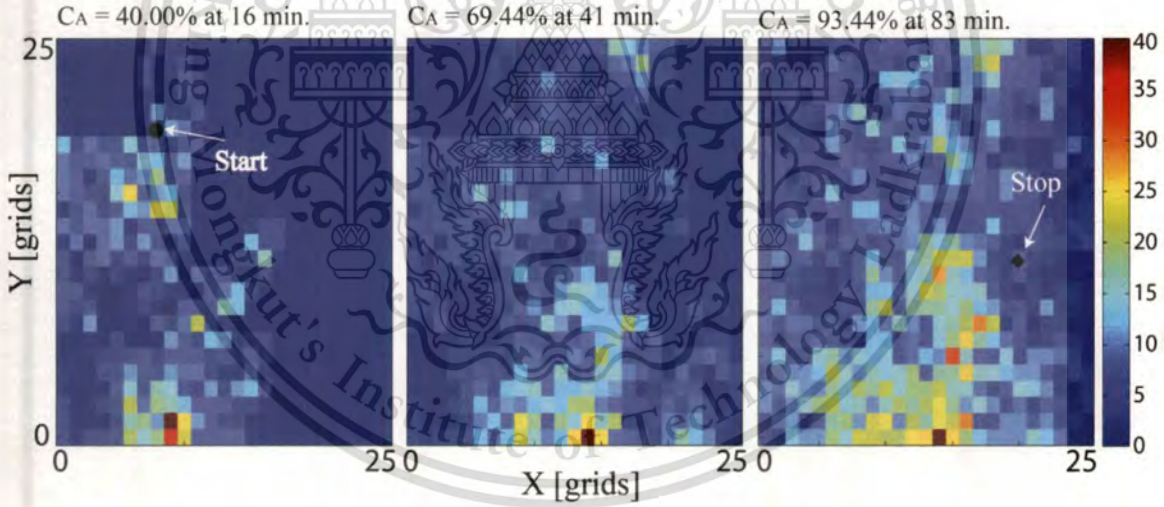


Fig. 4.11 The area coverage of the embedded chaotic mobile robot generated by Lorenz's equation where the initial point starting on quadrant Q2

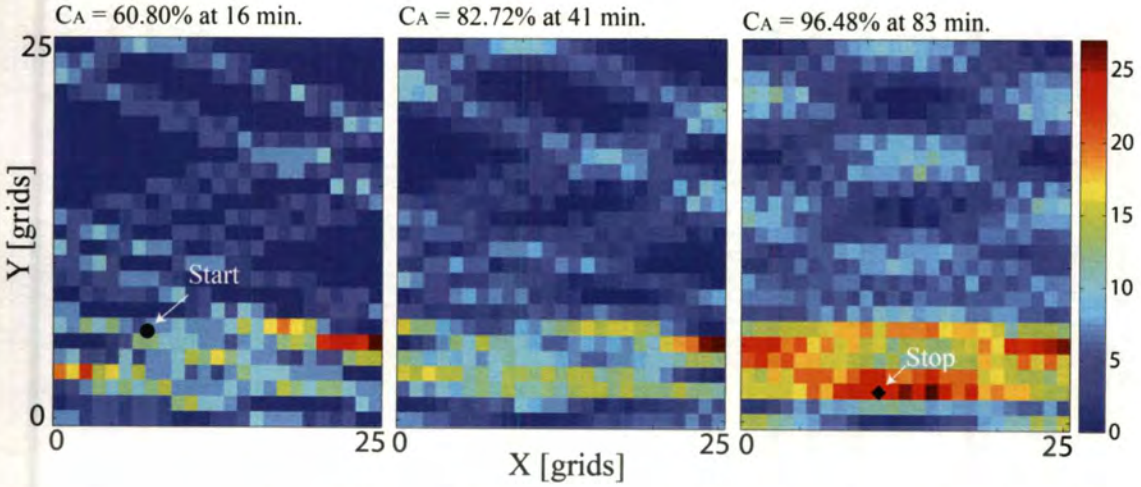


Fig. 4.12 The area coverage of the embedded chaotic mobile robot generated by Arnold's equation where the initial point starting on quadrant Q3

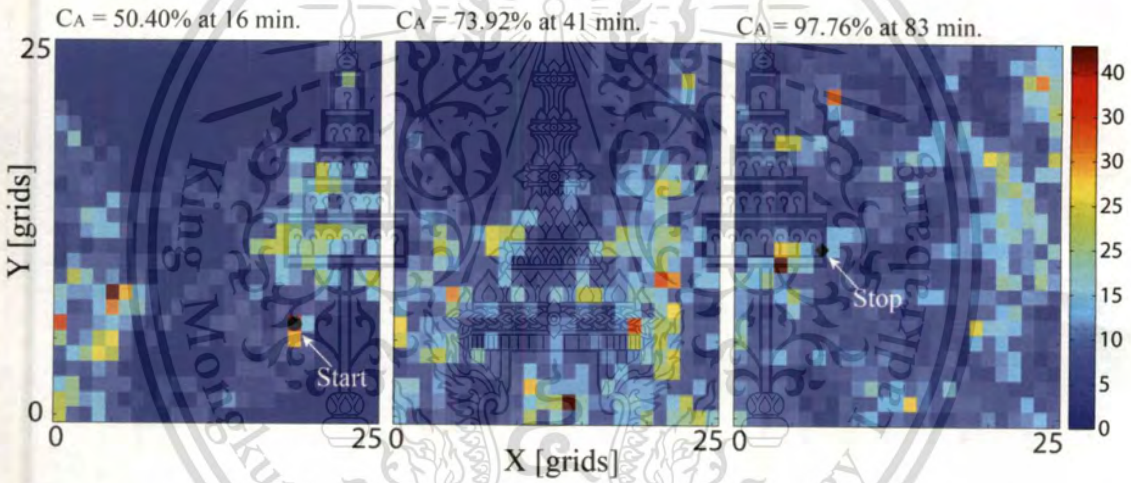


Fig. 4.13 The area coverage of the non-embedded type for Robot 3 in the scenario P_5

To validate the simulation results, an experiment is required to convince that the new method practically works for generating chaotic behavior that arises from the non-embedded complex robotic network. For this reason, the next chapter will present realization of the experiment.

Chapter 5

Experiment on the Complex Robotic Network

5.1 Introduction

One of the goals of this research is to develop a new model of a chaotic robotic network without embedding a chaotic system of equations into the robots controller. In Chapter 4, the non-embedded complex robotic network based on the inverse square law of distance was implemented by the 2-D simulation. It is numerically proved that the system is able to generate the chaotic behavior of the robot network. Accordingly, realization of the experiment is required to validate the solution in the simulation and to confirm the proposed.

This chapter deals with experimental setup and results of the non-embedded chaotic robotic network as well as its analysis. Firstly, the hardware including the robots, their sensing system and communication system are explained. Secondly, the implementation of the mathematical model on the non-embedded robotic network is described. Finally, the experiment results of the non-embedded robotic network are presented using the time series analysis, to prove the system in hand having the state of chaos.

For outline of the discussion, the behavior of complex networks have been discussed from botnets as threats on the cyberspace presented in Chapter 3. The mathematical and conceptual model of complex robotic network behaves chaotically in the real-space can be seen in Chapter 4. The experiment implementation of the non-embedded complex robotic network is conducted in this chapter to verify the proposed model.

5.2 The Non-Embedded Mobile Robot Hardware

A non-embedded complex robotic network comprises four mobile robots. The robot is implemented in a platform of two-wheeled mobile robot (2-WMR), and it is equipped with a very simple communication system based on infrared light sources. To cope with a collision against the wall and other robots, three mechanical bumpers are installed in the front side of each robot. The brain of the robot is supported by *PSoC*® microcontroller *Cypress CY8C29446-24PX1*. Figure 5.1 shows the four two-wheeled mobile robots used in this experiment. Building blocks of the robot include: the sensing system that consists of sensor devices and Analog to Digital Controller (ADC), mechanical bumpers for triggering the interrupt routine, controller software, the Pulse Width Modulation (PWM) for controlling the DC motor speed, and the motor driver. Figure 5.2 shows the block diagram of the building block of the robot.

Fig. 5.1 Four two-wheeled mobile robots for the non-embedded robotic network

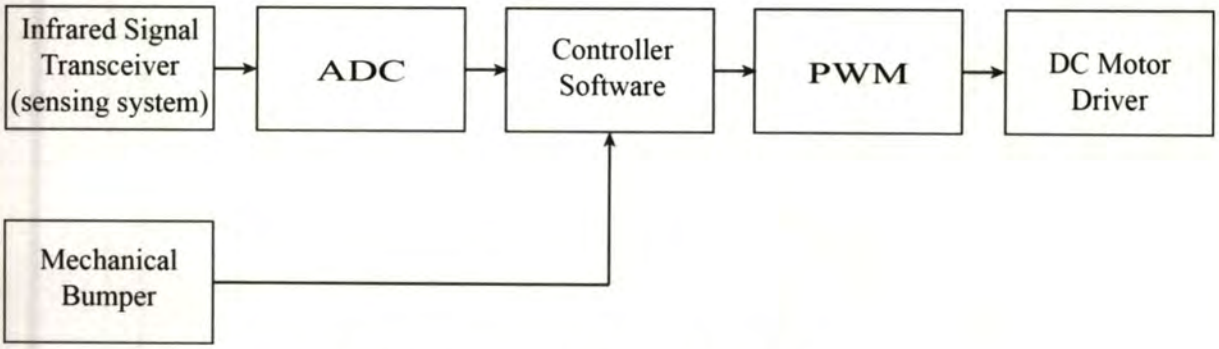


Fig. 5.2 Functional diagram of the robot

5.2.1 Two-Wheeled Mobile Robots

Four mobile robots were built in this experiment. They are in circular shape with the largest diameter 0.20*m* and height 0.15*m*. Each 2-WMR is made of an acrylic base, raised 0.75cm off the ground by two independently driven wheels and two balance metal half sphere as a support.

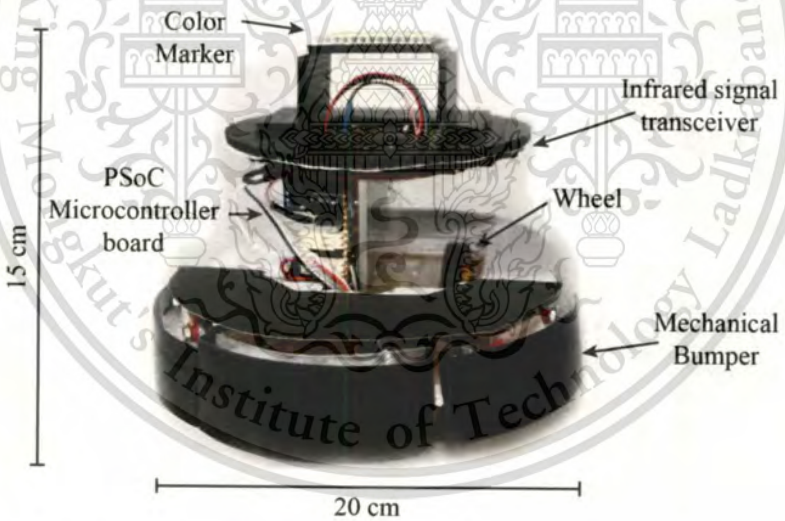


Fig. 5.3 Two-wheeled mobile robot used in the experiment

Figure 5.3 shows a single 2-WMR used in this experiment. Under the base, there is an 8.5V rechargeable battery pack, and two DC motors. A control circuit board stands up at the right angle as shown in the figure. The upper level is in a circular shape disc having 0.12*m* in diameter equipped with infrared LED transceivers. The top structure has a piece of plastic in circular shape with color markers for visual

tracking. The bumpers are mounted 0.05*m* up over the ground at the front side of the robot.

5.2.2 Mechanical Bumper

Three mechanical bumpers are installed at the half circle of the robot's body. These bumpers are used to sense the collision with other robots or with a wall. Each bumper has a 0.08*m* curve shape made of plastic. The bumper is supported with a screw, which has a 0.05*m* length attached on the acrylic platform. A rubber is used to make the bumper bouncing back after hitting the obstacle.

There are three bumpers in front of the robot; each bumper is set at the left, the right and the center. The center one senses an obstacle at the front side of the robot. For each left and right obstacles are sensed by the left and the right bumpers, respectively. The inner edge of the bumper will activate a limit-switch when the robot is hitting the obstacle. Once the limit switch is triggered, the interrupt sends the signal to the controller. Remark that the interrupt is a mechanism in a microcontroller to interrupt the running process. If it is activated, then the interrupt routine will be executed. This makes the robot turned aside from the obstacle depending on which side of a bumper is triggering. Figure 5.3 shows bumpers position at the robot.

5.2.3 Infrared Signal Transceiver (the Sensing System)

Recall the concept mentioned in Chapter 2 inspired by the interaction system of the robot network, the interaction obeys the inverse square law of distance among the robots. The physical phenomena in the real world such as sound wave, light, and radio wave obey the inverse square law as well. This experiment emphasizes on the inverse square law itself. For this reason and simple

consideration, this experiment utilizes infrared LED transceivers to exchange the interaction signal among the robots.

The robot recognizes the existence of the others by sensing the infrared light received at its photodiode sensor. The platform of infrared LED transceivers is designed as a circular shape with the 0.12*m* diameter, and it is mounted at 0.11*m* measure from the ground on the top of the robot. There are sixteen infrared LEDs as the transmitters and eight photodiodes as the receivers around the circle. The circular shape is designed to support the transmission infrared signal in omni-direction. Figures 5.3 and 5.4 show the installation and the design of the platform of the transceivers, respectively.

The LEDs continually emit the infrared light, and the photodiodes convert the received infrared signal into the voltage. The infrared LED transmitter emits signal at 40 *k*Hz frequency and 50% constant duty cycle. After the photodiodes receive infrared signal from other robots and convert it into the voltage, the ADC captures the voltage from the sensor driver every 10*ms*. Each robot uses the ADC on chip with a 10 *bits* resolution and the maximum voltage received by the infrared sensor is observed around 4V. Consequently, the 4V voltage is digitized to 1024 levels. Figure 5.5 shows how the observed voltage is digitized to the discrete levels of ADC based on the distance between sources and receivers.

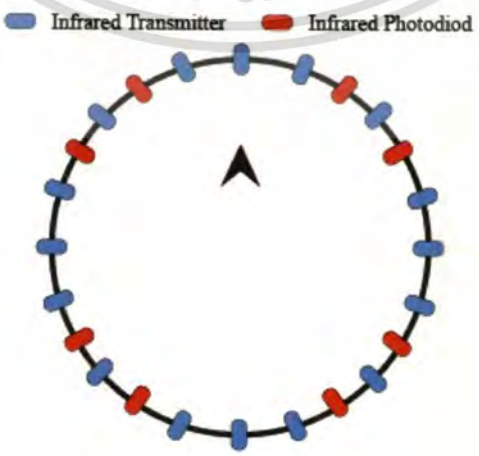


Fig. 5.4 Design diagram of infrared transceivers in a 12-cm diameter circle

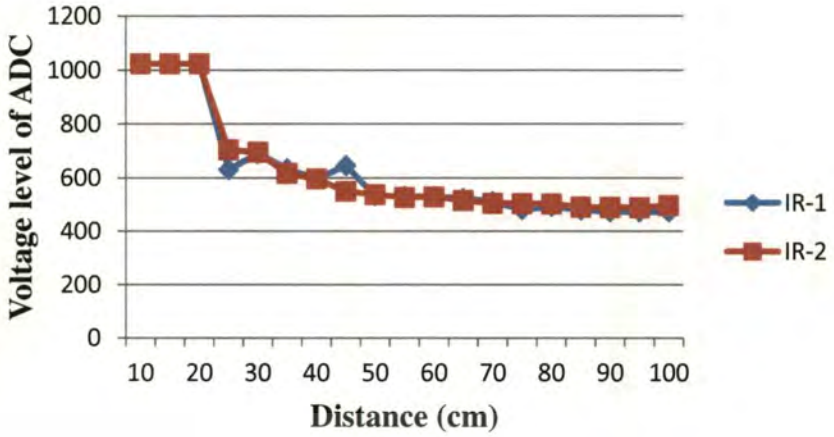


Fig. 5.5 Characteristics of the infrared photodiodes

5.2.4 PWM and DC Motor Driver

Two wheeled mobile robot herein adopts a differential drive platform. In this experiment, the right and the left wheels of the robot are driven by two PWMs. The right PWM serves for the right wheel, likewise for the left. The outputs of both PWMs are defined by the controller software.

The PWM generates a pulse train of square wave to control the DC motor. If both PWM outputs have the same duty cycle, then the DC motor will run approximately at the same speed. This makes the robot moved straight.

5.3 Robot Controller Software

The controller software is the main part that distinguishes between the embedded and non-embedded chaotically robotic network. In the embedded chaotic mobile robot, run time of the microcontroller is burdened by the computation of chaotic equations, localization and path planning algorithm. In contrast, the non-embedded chaotic robotic network can be achieved much simpler without any high cost computation.

In this experiment, to drive the left and the right wheels of the robot, the controller software is implemented by using the system in Eq. (4.4). Interpretation of Eq. (4.4) is simple by these assumptions:

1) s_i is the entity of Robot i;

2) $-ws_i$ is a degree of coupling strength that the Robot i interacts with the others in the workspace. It can be represented as a default constant velocity that makes the robots keep moving straight forward indicated there are no other robots interacting with. The constant velocity is manually tuned at $0.1m/s$. In practice, the $0.1m/s$ is reached by tuning both the left and the right PWM duty cycles around 50%.

3) $g\sum_{\substack{j=1 \\ i\neq j}}^{N}\frac{1}{r_{ij}^2}$ is the information signal transmitted from other robots and received by Robot i. This part will drive the robot to turn right or turn left depending on the interaction strength. In the realization, four photodiodes on the right side will sense the information signal come from the right, and four other photodiodes will serve for the left. The information signals received from the right are summed, likewise for the left. If the information signal on the right is stronger than the left, then the robot will turn right for 10ms duration with increasing the duty cycle to 75% of the left PWM. The left wheel's velocity is around 0.15m/s.

4) The last two parts of Eq. (4.4), $\dot{x}_i = v\cos(s_i)$ and $\dot{y}_i = v\sin(s_i)$, are equivalent to the right and the left PWM. The center of mass of the robot trajectory is located at the point (x_i, y_i), changed by the summation of Assumptions 2) and 3) equivalent to changing s_i. Similarly, the PWM is fed by the output of Assumptions 2), when there is no interaction.

These assumptions can be represented as a flow diagram. Figure 5.6 shows the algorithm of the controller software described as the flowchart below.

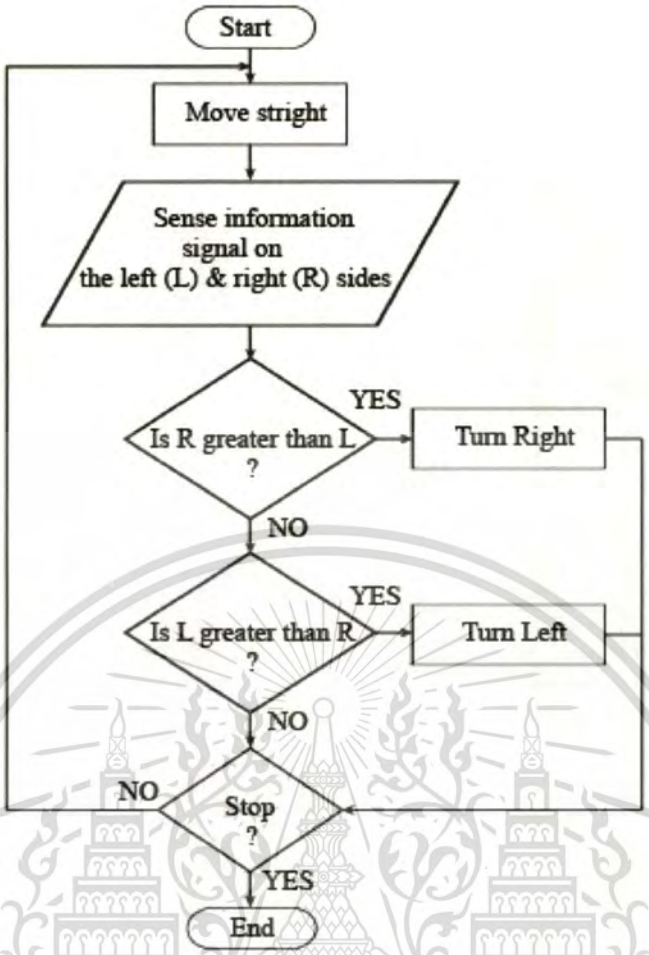


Fig. 5.6 The flowchart diagram of the controller software

5.4 Experimental Results

This section presents experimental results with two configurations. Firstly, a robot travels on the $1.5 \times 1.5 m^2$ workspace without any other robots around. Secondly, a non-embedded robotic network consists of four robots run inside the workspace. Six scenarios, which are different in the initial positions, are set as similar to the simulation as mentioned in Chapter 4. The camera capturing trajectories of the robots is installed at the ceiling. Figure 5.7 shows the experimental setup of the workspace and the monitoring camera.

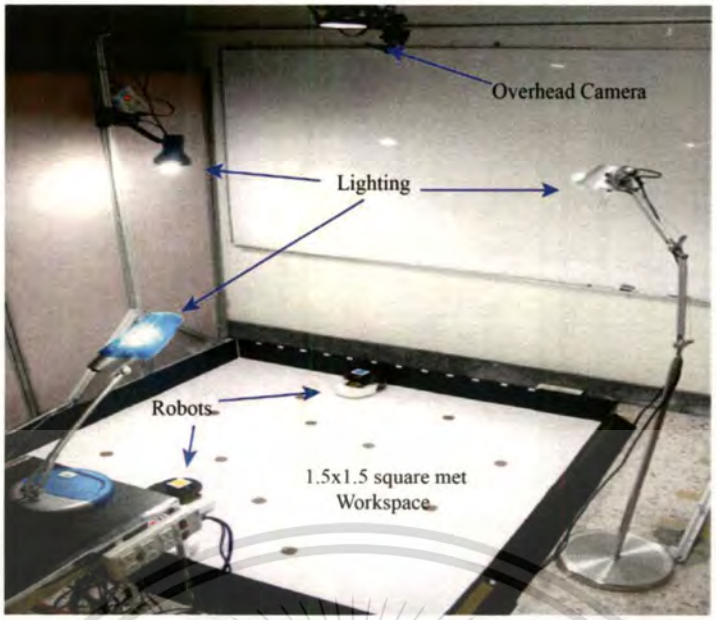


Fig. 5.7 The experimental setup of the workspace and the monitoring camera

5.4.1 Analysis of the Behavior of the Non-Embedded Single Mobile Robot

A mobile robot moves inside the field of workspace will behave like a bouncing ball. Because there are no other robots around, then the robot only moves with a constant velocity. The movement is also obeyed the mirror mapping technique when the robot is facing the wall. Figure 5.8 shows the trajectory captured with the camera. Noises are generated from the template matching process during when the captured frames mapping.

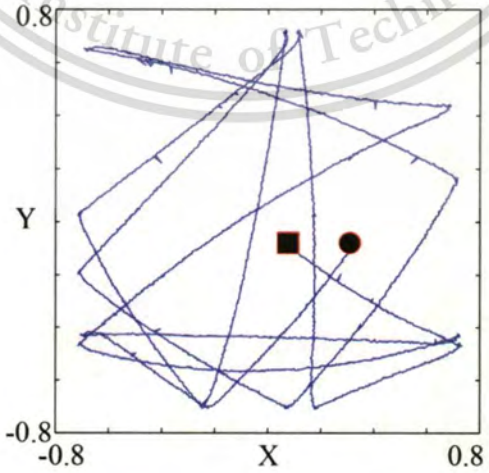


Fig. 5.8 The trajectory of the non-embedded single mobile robot where the robot starts moving from a square mark and ending at a circle mark

This scenario aims to confirm a statement in (4.4) that *ws* is a coupling parameter of the Robot *i* to interact with other robots and its environment. In case of no robots around, It can be interpreted as a movement with a constant velocity. Figure 5.8 shows the evidence that the model is work. Six sequences of time frames of photographs are shown in Fig. 5.9.

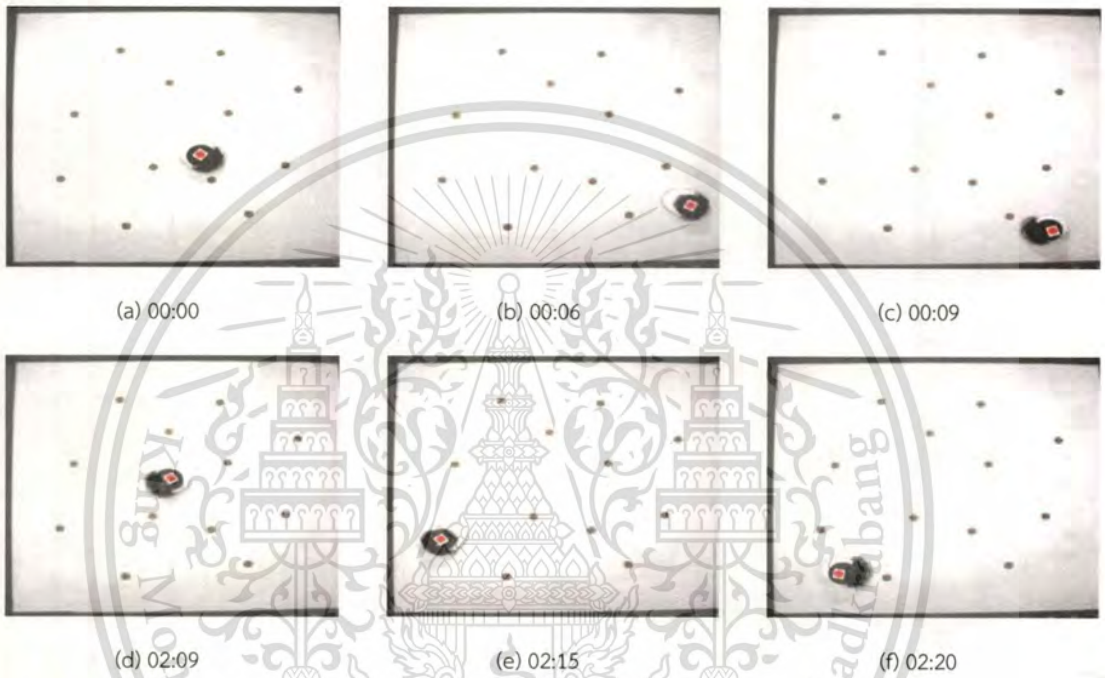
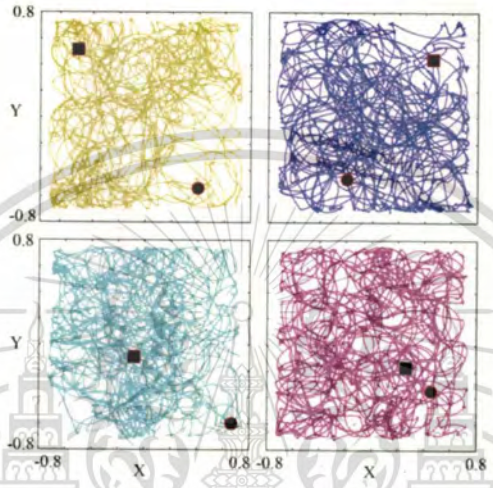


Fig. 5.9 Six snapshots of a robot move inside the workspace; Sequences (a)-(c) are snapped at the 1th minute and (d)-(f) are snapped at the 2th minute

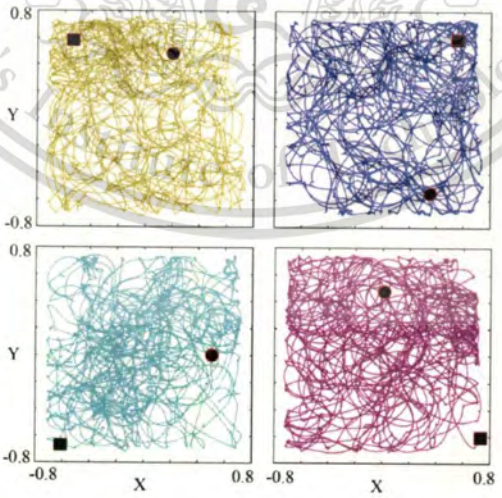
5.4.2 Analysis on the Behavior of the Non-Embedded Complex Robotic Network

The analysis is focused on the non-embedded robotic network comprises four robots, moving inside the workspace while interacting to each other. This experiment investigates the chaotic behaviors of the non-embedded robotic network. Trajectories of the robots in the $1.5\times1.5m^2$ workspace are mapped into the *Cartesian* coordinate with the linear transformation method mentioned in Chapter 2. In doing so, time series of robots' trajectories can be collected for further analysis.

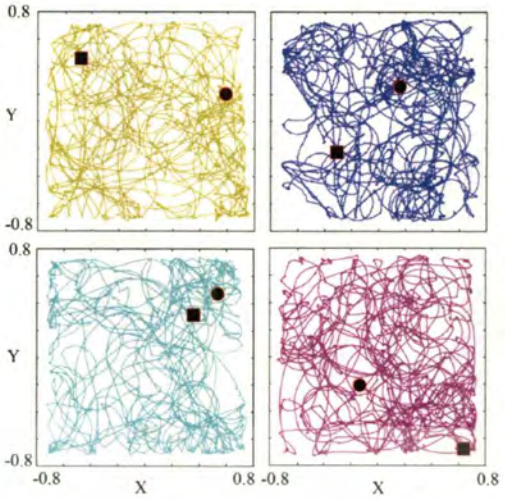
This experiment has six scenarios different in the initial position. For each scenario, robots travel inside the workspace for 20 *minutes*. Figure 5.10 shows the sequences of trajectories in the *X-Y* axes for all scenarios (P_1-P_6). The initial position for each robot is marked as a square and is ended marked by a circle. Figure 5.11 shows snapshots of the experiment in scenario P_1 on the sequence of time.



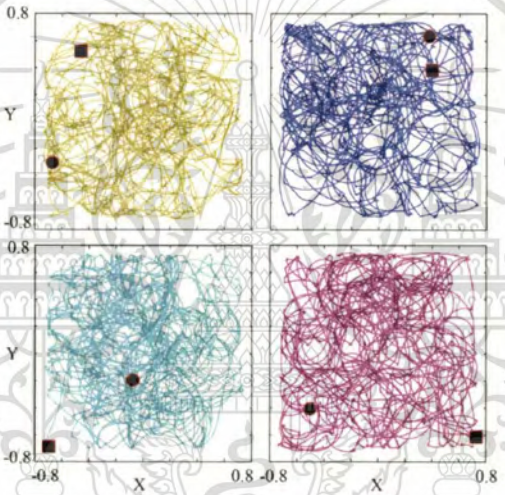
5.10-(a) Trajectories of P_1 of Robot "i" beginning with that of Robot 1 at the right-upper corner counted in anti-clockwise fashion



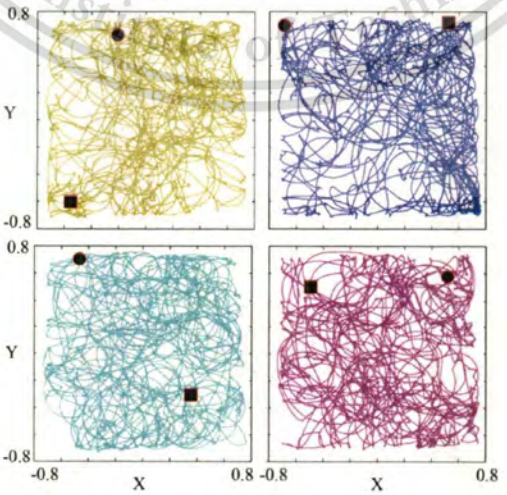
5.10-(b) Four robot's trajectories of P_2



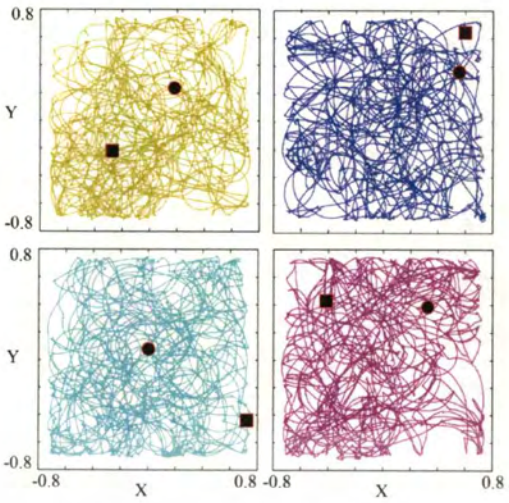
5.10-(c) Four robot's trajectories of P_3



5.10-(d) Four robot's trajectories of P_4



5.10-(e) Four robot's trajectories of P_5



5.10-(f) Four robot's trajectories of P_6

Fig. 5.10 Trajactories of the non-embedded complex robotic network for all scenarios P_1-P_6, which are represented in (a)-(f), respectively

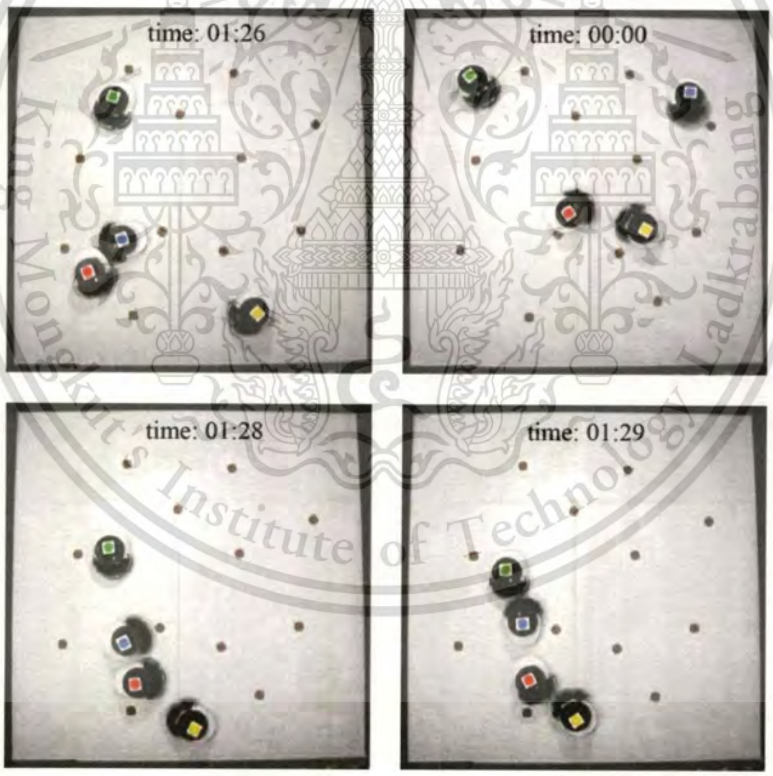


Fig. 5.11 Snapshots of the experiment in P1 on the sequence of time as shown in anti-clockwise direction

The non-embedded robotic network behaves chaotically in all six different scenarios as shown in a series of subfigures of Fig. 5.10. Those are evident that a strong interaction among robots and their environment plays the important role in

the emergently chaotic behaviors in the robotic network. As law of physics implied, this evidence also confirms that the most right part of the system in (4.4) $g\sum_{\substack{i\neq j\\ j=1}}^{N}\frac{1}{r_{ij}^2}$ is correctly represented the interaction model of the complex robotic network. Figure 5.12 shows trajectories resulted from the simulation and experiment visually look similar. The initial position of the robot is indicated by using a square mark. Similarly, a circle mark indicates the end of simulation. As shown in the figure, the trajectories have high similarities such as long curves, small curves, sharp curves, and covered the whole workspace.

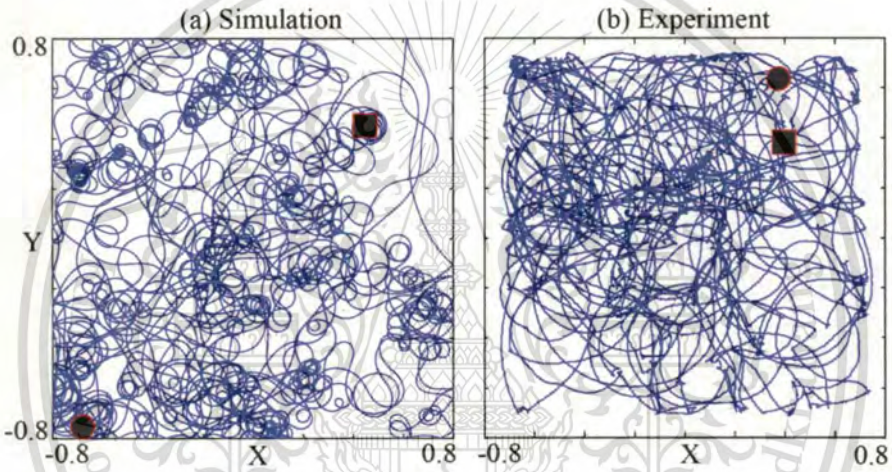


Fig. 5.12 Trajectories of non-embedded chaotic robotic networks: (a) generated with simulation of the system in (4.6) and (b) generated from experiment

The well-known method for identifying the chaotic system is the Largest *Lyapunov* exponent. A positive value of LLE indicates chaos [73]. Calculating LLE from the experimental data is more difficult than that data obtained from the ODE model. Firstly, the data quality, unwanted noises have been added to the data such as noises from image processing during capturing the trajectories. This is perhaps due to the low quality of the lighting system. Consequently, to cope with the data quality issue, a noise reduction method can be applied. Secondly, for calculating the LLE from the experimental data cannot be directly obtained as the calculation of the ODE model. In addition, phase space reconstruction with time-delay coordinate [50-

54] and an appropriate minimum embedding dimension [57] are required. All detail explanation of the method to calculate the LLE from the experimental data can be found in Chapter 2.

The LLEs from time series of the non-embedded robotic network in six scenarios are calculated by using Wolf algorithm [58] available in the form of user friendly GUI written by Perc [74, 75]. Similarly, time series from the simulation is treated as the experimental data to get the LLEs. Finally, LLEs from both experiment and simulation are compared.

The minimum embedding dimension is needed to calculate the LLEs from the experimental data. *Coa's* method [57] based on false nearest neighbors is used to estimate the minimum embedding dimension. For each scenario, we pick up time series from a robot and then calculate the minimum embedding dimensions. The calculation results of the minimum embedding dimension show the similar value for all scenarios. Figure 5.13 shows the minimum embedding dimensions estimation with *Cao's* method [57]. The curves of E1(*d*) for all scenarios versus dimension *d* saturated in the *d* is equal to 4, which can be seen as a knee on the graph. Thus, the minimum embedding dimensions equal 4 is used to feed the calculation of the LLEs with *Wolf's* algorithm [58]. Table 5.1 presents the LLEs of the non-embedded complex robotic network for both simulations and experiments.

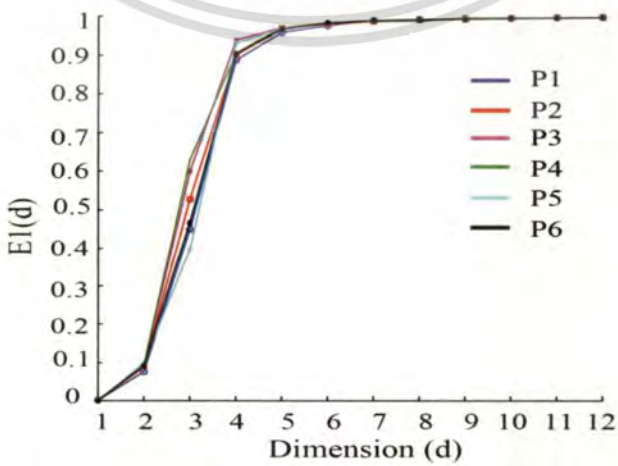


Fig. 5.13 The mean value of all the number of nearest neighbors E1(*d*)

Both simulation and experiment show the average (Avg) of the LLEs are positive, as shown in Table 5.1. Hence, this means that it is chaos. This average is taken from the number of time replacements from x_{it_n} to $x_{it_{n+1}}$, which are varied from 5% to 10% [58] of the total evolution. The average of the LLEs of the simulation for each scenario varied from 0.045 bits/s through 0.07 bits/s, and the average of all scenarios is 0.05 bits/s. The experiment reports that averages of the LLEs for each scenarios lay down between 0.084 bits/s and 0.12 bits/s, and the average for all scenarios is 0.099 bits/s.

Table 5.1 The LLEs of the non-embedded complex robotic networks: simulation versus experiment

Robot at scenario P_n	Simulation		Experiment	
	Range bits/sec	Average (Avg) bits/sec	Range bits/sec	Average (Avg) bits/sec
P_1 at Robot 1	0.06-0.07	0.07	0.10-0.13	0.12
P_2 at Robot 1	0.05-0.07	0.063	0.10-0.15	0.11
P_3 at Robot 1	0.03-0.06	0.05	0.09-0.11	0.10
P_4 at Robot 1	0.04-0.06	0.05	0.06-0.12	0.089
P_5 at Robot 3	0.03-0.06	0.045	0.07-0.10	0.084
P_6 at Robot 3	0.03-0.05	0.047	0.07-0.10	0.095

The difference in the LLEs between the simulation and experiment is 0.04 bits/s. Clearly, the LLE of the non-embedded robotic network on experiment is a little bit larger than the simulation. There are some reasons can be explained. Firstly, noise comes from environment adding more randomness to the trajectory data. Particularly in the experiment, the robot utilizes infrared transceivers to communicate, and thus it has some risks of noises generating from the ambient such as lighting and light reflection from the wall. Secondly, the robot does not have any

mechanism of controlling speed, which is designed as simple as possible with manually tune. Consequently, it gives a contribution of perturbation that comes from differential speed of the left and the right of motors that drive the robot. Next section concludes the discussion.

Chapter 6

Conclusions and Future Work

In this chapter, Section 6.1 presents the conclusion of this dissertation. Section 6.2 gives recommendations for the future work.

6.1 Conclusion

This research contributes in the field of complex networks and robotics. Two types of robots formed complex networks are under investigation and analysis: botnets on the Internet and mobile robotic network. For botnets, the bots propagates through cyberspace and the networks are formed physically by inherent interconnect of the Internet structure in real space. For mobile robots, the robots navigate through real space while forming a complex network via communication in information space or cyberspace. Both cases focus on behaviors of botnets and the robotic network.

Firstly, for the study of botnets, the CCC Datasets for 2009 and 2010 comprised the access logs of botnets' attacks, collected by 94 independent honeypots since May 1, 2008 through May 31, 2010 on the Japanese tier-1 backbone, are investigated. The *PrefixSpan* data mining algorithm is implemented to reveal the sequential behavior of coordinated pattern of botnet attacks. The CCC Dataset is filtered to fit the input data of the *PrefixSpan* method, and then the *PrefixSpan* discovers frequent sequential attack patterns based on the sequences of malware downloaded by honeypots.

The sequential attacks 3-pattern is further investigated based on IP-address and timestamps in order to understand the attack behavior of the botnet. In doing so, this research gives a classification and characteristics of the sequential attack

pattern of malware in the botnet. Several groups of source IP address of sequential attacks 3-pattern of malware are classified by using symbols $A_1,A_2...,A_5$ for representing *IP pattern code*, and then *IP pattern* of the sequence of source IP address of malware attacks are defined by using symbols S_1,S_2 and S_3 associated to the source IP address, as shown in Table 3.4. Another classification is based on the malware downloaded timestamps. The name is similar to the source IP address. Symbols E_1, $E_2...,E_4$ represent the time pattern code, meanwhile the timelines of honeypots downloading the malware are marked with symbols T_1,T_2 and T_3 , as described in Table 3.5. For example, if pattern $P_{3.242}$ is of type A_3E_3, as shown in Table 3.6, the first and third malware items are downloaded from the same source IP address (A_3), and the second and third malware items are downloaded at the same time (E_3). The investigation from two groups of attacks, which are shown in Table 3.6, we found that some attack patterns come from a unique host and some patterns attack from different hosts. Figure 3.5 illustrates how the coordinated attacks work.

Classification of sequential attack patterns based on time duration of attacks gives great information about how the botnet attacks behave in a certain time period. This classification shows the coordinated attacks are performed by multiple sequential patterns within a short period. Further investigation is conducted in two categories based on the malware sequence, which are called duplicate—it has the same malware appearing more than once in it—and nonduplicate. In particular, we discover the duplicate attack patterns are distributed uniformly within a year, and some malware comprise those pattern has the ability to disable some services on systems running Windows 2000 and Windows XP such as *Internet Connection Firewall* and *Internet Connection Sharing*. They listen to various ports and connect to an IRC server, and their potential for damage and propagation is rated as medium to high [69]. Regarding the botnet attack, we conjecture that the distribution diagram shown in Fig. 3.6 can be considered a distribution of the C&C activity of a botnet system. However, in case of nonduplicate attack patterns, they attacked intensively

for a short period, less than a month in a whole year, and the frequency of attack in a day is greater than the duplicate sequential attack 3-pattern. In addition, entropy analysis helps for discovering the most common sequential attack patterns involved in coordinated attacks. The entropy states about the probability of a particular sequential attack pattern attempts to attack all honeypots under investigation.

In the study of botnet, we reveal several behaviors that are useful for alerting Internet users against botnet attack threats. A vigilance and sensitivity against threats of botnet attacks can be established by understanding the behavior of the spread of malware from its source IP address and malware downloaded timestamps. Source IP addresses of malware play the important role in the investigation of botnet attacks. Some malware are sent by botnet through a unique source IP address and others from multiple of source IP address. Equally important, the botnet attacks are performed with multiple sequential attack patterns within a certain period—less than a month. The evidence, which are explained above concludes that the botnet employs a collaboration and coordination strategy to attack victims.

Secondly, for mobile robots, the new model of the non-embedded complex robotic network is proposed. This robotic network aims to substitute the embedded chaotic robots for generating chaotic movement. Traditionally, chaos has been implemented on a single and multiple chaotic mobile robots by programming chaotic equations into the robots' controllers. However, this method has a drawback in fixed chaotic patterns, self-localization and path planning. This research proposes a method for chaotically robotic network by implementing a simple interaction among robots. Four two-wheeled mobile robots compose a complex robotic network. Their interactions (links) are dynamically changed obeying the inverse square law of distance. The model is represented as the first Order Differential Equation (ODE) dynamical system. The fourth-order Runge-Kutta method is employed for numerical computation. The simulation results show the complex robotic network behaves chaotically due to high interaction among robots. Indeed the robotic network cannot

preserve the chaotic behavior when they lost their interactions. The proposed model for non-embedded complex robotic network is confirmed that it has property of chaos that is indicated by the positive value of the largest Lyapunov exponent. On performance comparison with the embedded counterparts, the proposed non-embedded complex robotic network is evaluated by exploring unknown terrain. The proposed scheme provides better area coverage than the embedded Lorenz's and Arnold's systems.

Thirdly, the experiment implementation is developed to validate whether the proposed non-embedded complex robotic network is practically and correctly performed. The robot's structure is a two-wheeled mobile robot (2-WMR), and it is equipped with a very simple communication system based on infrared light. To cope with congestion against the wall and other robots, three mechanical bumpers are installed in the front side of each robot. The brain of the robot is supported by PSoC® microcontroller CY8C29446-24PX1 by Cypress. Each robot behaves very simple; the robot moves forward to a constant speed when there are no signals from other robots. The robot turns right or left depending on where the strongest interaction comes from. Four 2-WMRs travel inside the $1.5\times1.5m^2$ workspace, and the overhead camera captures trajectories of the robots run underneath. In a simple case, a mobile robot moves inside the workspace behaved like a bouncing ball because there are no agents around. In the other case, the four robots move inside the workspace while interact to each other creating chaotic trajectories. The visualization of chaotic trajectories from both simulation and experimental results look similar. The largest Lyapunov exponent estimated from the time series data in the experiment shows positive and is greater than zero. This indicates that the system is chaos.

In summary, behaviors of two kinds of robots forming complex networks are investigated both in the cyberspace and the real space, which are represented by the botnet and the complex robotic networks, respectively. We are interested in

behaviors of the complex networks in term of characteristics of the behaviors: dead fixed point, periodic, or chaos. We found in the case of cyberspace that the sequential attack patterns of malware in the botnet are an example of the periodic behavior of complex network. For real space, in case of no other agents around, a robot would move with constant velocity and the case corresponding to a phenomenon of steady state in complex network. In contrast, a phenomenon of chaos can happen when complex robotic network has high interactions. All above phenomena lead to the conclusion that they come from a deterministic law. In the establishment of botnet on the Internet, the attacker or a programmer of malware *systematically develops* a specific malware with special function to work in coordinated manner and in sequence. Likewise, in simulation for the case of non-embedded complex robotic network, each robot is *programmed* to have the inverse square law interaction. Note that terms of "systematically develop" and "program", mentioned in italic words, are evidence of the complex network comprises deterministic system elements. Undoubtedly, the deterministic system can establish chaos when the interaction is beyond the threshold of linearity as evidence by strong interaction among the robots both in simulation and experimental results.

6.2 Recommendations for Future Work

6.2.1 The real time sequential attacks pattern monitor

Throughout this dissertation in particular at the botnet attack, we found some valuable information steams from the classification of the sequential attack pattern based on time duration of the attack. According to this research, attacks or infection occurred for a certain period of time around 20 to 25 days. At the zero day (first day of attack), the new sequential attack patterns appear continuously to establish the botnet. In this research, the information is achieved by mining the static dataset, which is collected by honeypots. It will be a challenge to achieve the information in the real continuous time. By doing so, the information can be used for alerting

people who are responsible for the network security and the defense. Moreover, it is possible to integrate the firewall system to improve the performance of blocking malicious activities.

6.2.2 Complex robotic network searching team

In this research, the non-embedded complex robotic network shows chaotic behaviors in a condition of high interaction strength among the agents. The predefined workspace is set physically to achieve this condition. However, this may not be exact the real world, in which the assigned workspace is a square shape with limited area. To enhance the system, a candidate method can be improved by enhancing performance of the communication system. The communication system with wide area coverage such as Wi-Fi or Wi-MAX is able to substitute the physical border in the real-world applications.

Searching and rescue teams in large unknown area need a lot of team members to work effectively. Searching in the unknown terrain with chaotic system guiding pattern has been studied in [17, 19, 40]. However, those studies are limited in a closed workspace. The advantage of the non-embedded chaotic robotic network is the chaotic behavior emerging from their interactions. For this reason, we recommend developing an expanding interaction competency to establish a virtual space based on the communication area coverage. And thus the workspace would become a flexible size.

Finally, the chaotic robot team for searching in the large unknown area can be developed. Every time a robot moves beyond a certain distant limit from the center of the network, the robot is programmed to heading back into the center of the network. Consequently, the chaotic behaviors of the robotic network can be preserved inside the virtual workspace. This concept is possible to be extended to chaotic searching along the predefined path, in which the chaotic robotic network will behave chaotically while move toward the target point.

References

[1] M. A. Rajab, J. Zarfoss, F. Monrose & A. Terziz [2006] "A multifaceted approach to understanding the botnet phenomenon," *in Proc. the 6th ACM SIGCOMM conference on Internet measurement*, pp.41-52.

[2] N. Provos and T. Holz [2007]. "Virtual Honeypots: From Botnet Tracking to Intrusion Detection," *Addison Wesley profesional*, pp.359-390.

[3] H. R. Zeidanloo and A. A. Manaf [2009] "Botnet Command and Control Mechanisms," *Computer and Electrical Engineering, ICCEE '09. Second International Conference*, pp.564 -568.

[4] B. McCarty [2003] "Botnets: big and bigger," *Security Privacy, IEEE,* vol. 1, pp.87-90.

[5] L. Spitzner [2002] "Honeypots: Tracking Hackers," *Addison Wesley Profesional*.

[6] P. Wang, S. Sparks & C. C. Zou [2010] "An Advanced Hybrid Peer-to-Peer Botnet," *IEEE Trans. Dependable and Secure Computing,* No.2, vol.7, pp.113-127.

[7] E. Hellweg [2004] "When botnets attacks," last accessed date July 18, 2011, available: http://www.technologyreview.com/computing/13771.

[8] C. Wilson [2008] "Botnets, Cybercrime, and Cyberterrorism: Vulnerabilities and Policy Issues for Congress," *CRS Report for Congress,* last accessed date July 18, 2011, available: www.fas.org/sgp/crs/terror/RL32114.pdf.

[9] G. Alvarez, F. Montoya, M. Romera & G. Pastor [2000] "Cryptanalysis of a chaotic encryption system," *Physics Letters A,* vol. 276, pp. 191-196.

[10] X.-y. Wang and Q. Yu [2009] "A block encryption algorithm based on dynamic sequences of multiple chaotic systems," *Communications in Nonlinear Science and Numerical Simulation,* vol. 14, pp. 574-581.

[11] A. Akhshani, S. Behnia, A. Akhavan, H. A. Hassan & Z. Hassan [2010] , "A novel scheme for image encryption based on 2D piecewise chaotic maps," *Optics Communications,* vol. 283, pp. 3259-3266.

[12] L. Zhengguo, K. Li, C. Wen & Y. C. Soh [2003] "A new chaotic secure communication system," *IEEE Trans. Communications,* No.8, vol. 51, pp. 1306-1312.

[13] W.-D. Chang [2009] "Digital secure communication via chaotic systems," *Digital Signal Processing,* vol. 19, pp. 693-699.

[14] P. Sooraksa and K. Klomkarn [2010] ""No-CPU" Chaotic Robots: From Classroom to Commerce," *Circuits and Systems Magazine, IEEE,* vol. 10, pp. 46-53.

[15] C. Changkyu, S.-G Hong, J.-H Shin, I.-K Jeong & J.-J Lee [1995] "Dynamical path-planning algorithm of a mobile robot using chaotic neuron model," in *Proc. 1995 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pp. 456-461.

[16] L. S. Martins-Filho and E. E. N. Macau [2007] "Trajectory Planning for Surveillance Missions of Mobile Robots," *Studies in Computational intelligence (SCI),* vol. 76,pp. 109-117.

[17] Z. Ailin and H. Leung [2007] "Cooperation Random Mobile Robots Based on Chaos Synchronization," in *Mechatronics, ICM2007 4th IEEE International Conference on*, pp. 1-5.

[18] A. Jansri, K. Klomkarn & Pitikhate Sooraksa [2004] "Further investigation on trajectory of chaotic guiding signals for robotic systems," in *Proc. ISCIT*, vol. 2, pp. 1166-1170.

[19] Y. Bae [2004] "Target searching method in the chaotic mobile robot," *in Proc. of the 23rd IEEE Digital Avionics Systems Conference (DASC'04)*, vol. 2, pp. 12.D.7-12.12.1-9.

[20] K. Fallahi and H. Leung [2010] "A Cooperative Mobile Robot Task Assignment and Coverage Planning Based on Chaos Synchronization," *International Journal of Bifurcation and Chaos (IJBC),* vol. 20, pp. 161-176.

[21] Y. Nakamura and A. Sekiguchi [2001] "The chaotic mobile robot," *IEEE Trans. Robot. Autom.,* vol. 17, pp. 898-904.

[22] S. H. Strogatz [2001] "Exploring complex networks," *Nature,* vol. 410, pp. 268-276.

[23] J. C. Sprott [2008] "Chaotic dynamics on large networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 18, pp. 023135-023135-9.

[24] D. Geer [2005] "Malicious bots threaten network security," *Computer,* vol. 38, pp. 18-20.

[25] J. B. Grizzard, V. Sharma, S. Nunnery, B. B. H. Kang & D. Dagon [2007] "Peer-to-peer botnets: overview and case study," in Proc. of the first conference on First Workshop on Hot Topics in Understanding Botnets, pp. 1-1.

[26] N. Provos [2004] "A Virtual Honeypot Framework," in *13th USENIX SecuritySymposium*, San Diego, CA, pp. 1-14.

[27] M. Adeel, A. A. Chaudhry, E. Ahmed, K. Samad & N. M. Shaikh [2005] "Honeynets: An Architectural Overview," in *Engineering Sciences and Technology, 2005. SCONEST 2005. Student Conference on*, pp. 1-6.

[28] S. Staniford, V. Paxson & N. Weaver [2002] "How to Own the Internet in Your Spare Time," *in Proc. of the 11th USENIX Security Symposium*, pp. 149-167.

[29] K. Kuwabara, H. Kikuchi, M. Terada & M. Fujiwara [2010] "Heuristics for Detecting Botnet Coordinated Attacks," in *Availability, Reliability, and Security, 2010. ARES '10 International Conference on*, pp. 603-607.

[30] E. Cooke, F. Jahanian & D. McPherson [2005] "The Zombie roundup: understanding, detecting, and disrupting botnets," *in Proc. of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI'05),* last accessed date July 18, 2011 available: http://www.usenix.org/events/sruti05/tech/full_papers/cooke/cooke_html/

[31] N. Daswani, M. Manasse, M. Najork & J. L.Wiener [2007] "The anatomy of clickbot.a," in *In USENIX Hotbots'07*, pp. 10-10.

[32] G. Gu, P. Porras, V. Yegneswaran, M. Fong & W. Lee [2007] "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," *in Proc. of the 16th USENIX Security Symposium (Security'07).* No. 12.

[33] G. Gu P. Porras, V. Yegneswaran, M. Fong & W. Lee [2008] "BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet

detection," in *SS'08: Proceedings of the 17th conference on Security symposium*, Berkeley, CA, USA, pp. 139-154.

[34] O. Thonnard and M. Dacier [2008] "A framework for attack patterns' discovery in honeynet data," *Digital Investigation,* vol. 5, pp. S128-S139.

[35] W. Lu, M. Tavallaee & A. A. Ghorbani [2009] "Automatic discovery of botnet communities on large-scale communication networks," in *Proc. of the 4th International Symposium on Information, Computer, and Communications Security*, New York, NY, USA, pp. 1-10.

[36] H. Husna, S. Phithakkitnukoon, S. Palla & R. Dantu [2008] "Behavior analysis of spam botnets," in *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pp. 246 - 253.

[37] A. Jansri, K. Klomkarn & P. Sooraksa [2004] "On comparison of attractors for chaotic mobile robots," in *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, vol. 2, pp. 2536-2541.

[38] M. Monirul Islam and K. Murase [2005] "Chaotic dynamics of a behavior-based miniature mobile robot: effects of environment and control structure," *Neural Networks,* vol. 18, pp. 123-144.

[39] C. Chanvech, K. Klomkarn & P. Sooraksa [2006] "Combined Chaotic Attractor Mobile Robots," in *SICE-ICASE, 2006. International Joint Conference*, pp. 3079-3082.

[40] L. S. Martins-Filho and E. E. N. Macau [2007] "Patrol Mobile Robots and Chaotic Trajectories," *Mathematical Problems in Engineering,* Hindawi.

[41] "Chaos," in *Merriam-Webster Online Dictionary*, last accessed date July 18, 2011, available: http://www.merriam-webster.com/dictionary/chaos.

[42] S. H. Strogatz [1998] "Overview: Chaos, Fractals, and Dynamics," in *Nonlinear Dynamics and Chaos*, 1st ed: Persues Books Publishing, L.L.C., p. 2.

[43] A. A. Sharov [1996] "Equilibrium: Stable or Unstable?," last accessed date: July 18, 2011, Available: http://home.comcast.net/~sharov/PopEcol/lec9/equilib.html

[44] S. H. Strogatz [1998] "Chaos on a Strange Attractor," in *Nonlinear Dynamics and Chaos*, 1st ed: Persues Books Publishing, L.L.C., pp. 317-325.

[45] M. Lakshmanan and S. Rajasekar [2003] "Qualitative Features," in *Nonlinear Dynamics Integrability, Chaos, and Patterns*, ed: Springer.

[46] "Chaos Theory," last access date: August 9, 2011. Available: http://www.absoluteastronomy.com/topics/Chaos_theory

[47] G. L. Baker and J. P. Gollub [1990] "*Chaotic Dynamics an Introduction," Cambridge: Cambridge University Press*.

[48] Hilb and R. Corn [2000] "Chaos and Nonlinear Dynamics an Introduction for Scientists," *Oxford University Press*.

[49] J. C. Sprott [2003] "Chaos and Time-Series Analysis," *Oxford University Press*.

[50] J. Theiler [1986] "Spurious dimension from correlation algorithms applied to limited time-series data," *Phys. Rev. A,* vol. 34, pp. 2427-2432.

[51] J. Theiler [1987] "Efficient algorithm for estimating the correlation dimension from a set of discrete points," *Physical Review A,* vol. 36, p. 4456.

[52] J. Theiler [1990] "Estimating fractal dimension," *J. Opt. Soc. Am. A,* vol. 7, pp. 1055-1073.

[53] M. B. Kennel, R. Brown & H. D. I. Abarbanel [1992] "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Phys. Rev. A,* vol. 45, pp. 3403-3411.

[54] M. B. Kennel and S. Isabelle [1992] "Method to distinguish possible chaos from colored noise and to determine embedding parameters," *Phys. Rev. A,* vol. 46, pp. 3111-3118.

[55] A. M. Fraser and H. L. Swinney [1986] "Independent coordinates for strange attractors from mutual information," *Physical Review A,* vol. 33, p. 1134.

[56] H. Kantz and T. Schreiber [1995] "Dimension estimates and physiological data," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 5, pp. 143-154.

[57] L. Cao [1997] "Practical method for determining the minimum embedding dimension of a scalar time series," *Physica D: Nonlinear Phenomena,* vol. 110, pp. 43-50.

[58] A. Wolf, J. B. Swift, H. L. Swinney & J. A. Vastano [1985] "Determining Lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena,* vol. 16, pp. 285-317.

[59] R. Khare and A. K. Solutions [2007] "Wheeled Mobile Robots," in *Robotics*, ed New Delhi: Infinity Science Press. LLC., pp. 155-212.

[60] G. Novak and M. Seyr [2004]"Simple Path Planning Algorithm for Two-Wheeled Differentially Driven (2WDD) Soccer Robots.," in *Proceedings of the Second Workshop on Intelligent Solutions in Embedded Systems, WISES 2004*, Graz University of Technology, Graz, Austria, pp. 91-102.

[61] W. H. Press, S. A. Teukolsky, W. T. Vetterling & B. P. Flannery [2007] "Integration of Ordinary Differential Equations," in *Numerical Recipes The Art of Scientific Computing.*, 3rd ed Cambridge: Cambridge University Press, pp. 899-920.

[62] M. E. J. Newman [2003] "The Structure and Function of Complex Networks," *SIAM Review,* vol. 45, pp. 167-256.

[63] J. C. Sprott [2008] "Simple models of complex chaotic systems," *American Journal of Physics,* vol. 76, pp. 474-480.

[64] N. R. Rosyid and P. Sooraksa [2009] "Minimum Complex Network of Chaotic Robots," presented at the Asia Simulation Conf. JSST, Shiga, Japan. p. ID040.

[65] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal & M.-C Hsu [2001] "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth," In Proc. Int. Conf. Data Engineering (ICDE'01), pp. 215-224.

[66] F. Pedro "A Survey on Sequence Pattern Mining Algorithms" last accessed date: July 18, 2011, available: http://alfa.di.uminho.pt/~pedrogabriel/papers/SM_survey.pdf.

[67] R. Agrawal and R. Srikant [1994] "Fast Algorithms for Mining Association Rules," in *Proc. 20th Int. Conf. Very Large Data Bases*, J. B. Bocca, *et al.*, Eds., ed: Morgan Kaufmann, pp. 487--499.

[68] R. Agrawal and R. Srikant [1995] "Mining sequential patterns," in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pp. 3-14.

[69] Trendmicro [2007] "*Malware Threat Enciclopedia"* last accessed date: July 18, 2011, Available: http://about-threats.trendmicro.com/

[70] S. Boccaletti, V. Latora, Y. Moreno, C. Chavez & D.-U. Hwang [2006] "Complex networks: Structure and dynamics," *Physics Reports,* vol. 424, pp. 175-308.

[71] V. Latora and M. Marchiori [2003] "Economic small-world behavior in weighted networks," *The European Physical Journal B - Condensed Matter and Complex Systems,* vol. 32, pp. 249-263.

[72] P. Sooraksa, S. Miyamoto & H. Aoyama [1999] "A Low-Cost Passive Acoustic Resonator for Insect-Like Microrbotic Systems," in Proc. Automation and Robotics in Construction XVI, pp. 725-729.

[73] J. C. Sprott [200] "Strange Attractors," last accessed date July 18, 2011, Available: http://sprott.physics.wisc.edu/phys505/lect06.htm

[74] M. Perc [2005] "The dynamics of human gait," *European Journal of Physics*, vol. 26, p. 525.

[75] M. Perc [2005] "Nonlinear time series analysis of the human electrocardiogram," *European Journal of Physics*, vol. 26, p. 757.

Appendix A

Cartesian to Windows: Coordinates Conversion

In the experiment of this research, the overhead camera captures the robot's trajectories generated by robots run underneath. This process is offline, which means the movie is saved in a raw movie format. Originally, the display on Windows consists of a rectangular region that can be reconfigured in accordance with the user requirements. Therefore, we can convert all points in Windows coordinate into the Cartesian coordinate system.

In mathematics, we represent a point by using the Cartesian coordinate. Conversely, Windows output on the display screen comprises pixels that the coordinate system uses — the positive integer only. Yet, it doesn't mean a point with coordinates like (-2, 4.093) wich cannot be displayed on Windows screen. In order to display such a point on Windows screen, we need a transformation method so that this point maps to a pixel on Windows screen.

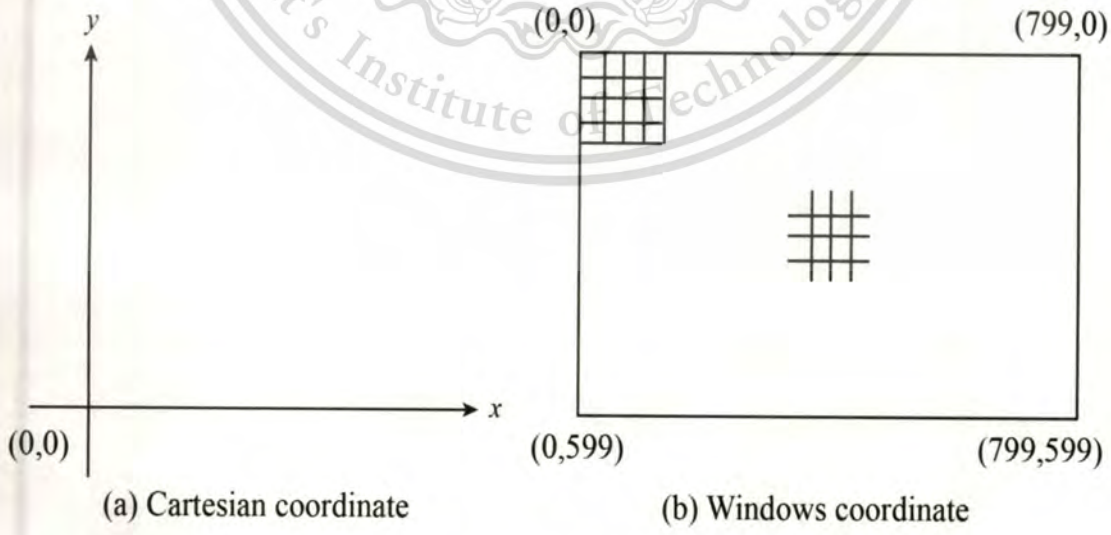


Fig. A.1 (a) The Cartesian coordinate system and (b) Windows coordinate system with 800×600 resolution.

Figure A.1 shows the Cartesian and Windows coordinate systems. A point in the Cartesian system is defined as (x,y) and as (X,Y) for a pixel in the Windows system. The Windows coordinate system starts at (0, 0) in the left-hand corner at the origin, and the resolution of the display monitor defines the end of the point in Windows. For example, the display monitor has 800×600 screen resolution, it means 800 columns and 600 rows of pixel with coordinates of (799,0), (0,599), and (799,599) on the top right the bottom left and the bottom right-hand corners, respectively.

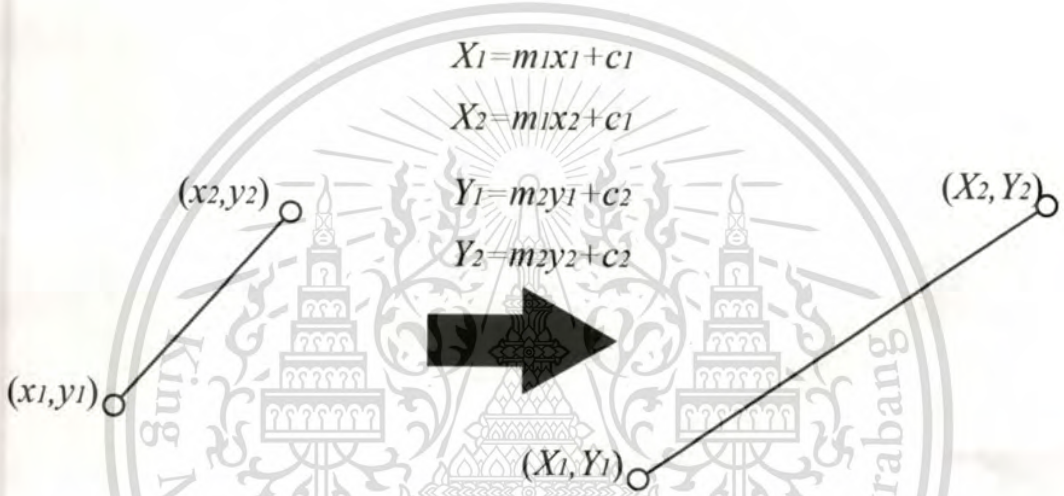


Fig. A.2 Conversion from Cartesian coordinate system into Windows coordinate system

Mapping a point (x,y) from the Cartesian system onto its corresponding pixel (X,Y) on the Windows is involving the linear relationships, $X=m_1x+c_1$ and $Y=m_2y+c_2$. Where m_1 and m_2 are the gradients in the mapping $x \rightarrow X$ and $y \rightarrow Y$, respectively. The constants c_1 and c_2 are the y-axis intercepts of the lines in the Cartesian system. Figure A.2 shows a linear transformation from a line in the Cartesian system to Windows. The line in the left is made up of the points (x_1,y_1) and (x_2,y_2), which maps to the Windows coordinates, (X_1,Y_1) and (X_2,Y_2), respectively. The mapping involving $x \rightarrow X$ is linear represented as follows:

$$X_1 = m_1x_1 + c_1, \qquad \text{(A.1a)}$$

$$X_2 = m_1 x_2 + c_1. \quad \text{(A.1b)}$$

At the same time, the mapping $y \rightarrow Y$ is also linear represented as

$$Y_1 = m_2 y_1 + c_2, \quad \text{(A.2a)}$$

$$Y_2 = m_2 y_2 + c_2. \quad \text{(A.2b)}$$

Solving the first two equations, we obtain

$$m_1 = \frac{X_2 - X_1}{x_2 - x_1}, \quad \text{(A.3a)}$$

$$c_1 = X_1 - m_1 x_1. \quad \text{(A.3b)}$$

The mapping equation in $y \rightarrow Y$ is solved in the same manner to get the results as follows:

$$m_2 = \frac{Y_2 - Y_1}{y_2 - y_1}, \quad \text{(A.4a)}$$

$$c_1 = X_1 - m_1 x_1. \quad \text{(A.4b)}$$

Appendix B

GUI Simulator for Complex Robotic Network

In this research, we simulate the non-embedded complex robotic network that is described in Chapter 4. The Graphical User Interface (GUI) is written in C++ with Microsoft Visual Studio 2008. We use the Microsoft Fundamental Class (MFC) framework for developing the simulator. For numerical calculation, we use the fourth order Rungge-Kutta method.

The GUI simulator is simple and is divided into two parts: main window and a dialog box. Main window provides user a "system" button to open the dialog box. The robotic network is simulated on the main window. Figure B.1 shows the main window.

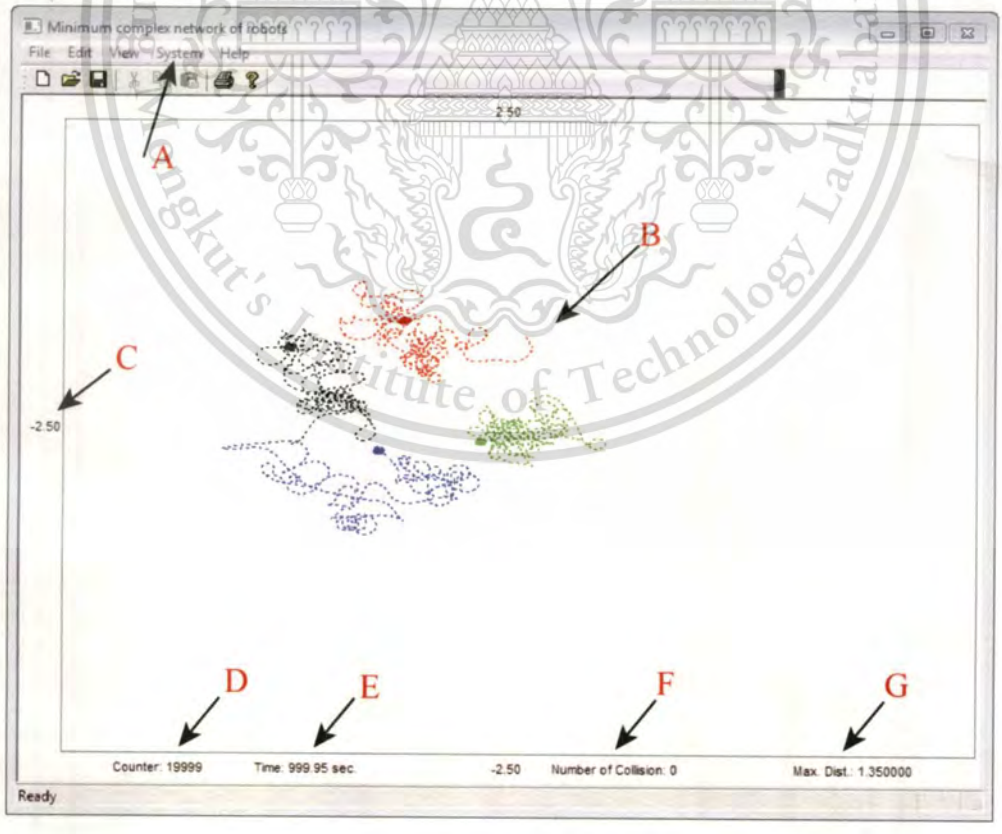


Fig. B.1 Main window of the GUI simulator

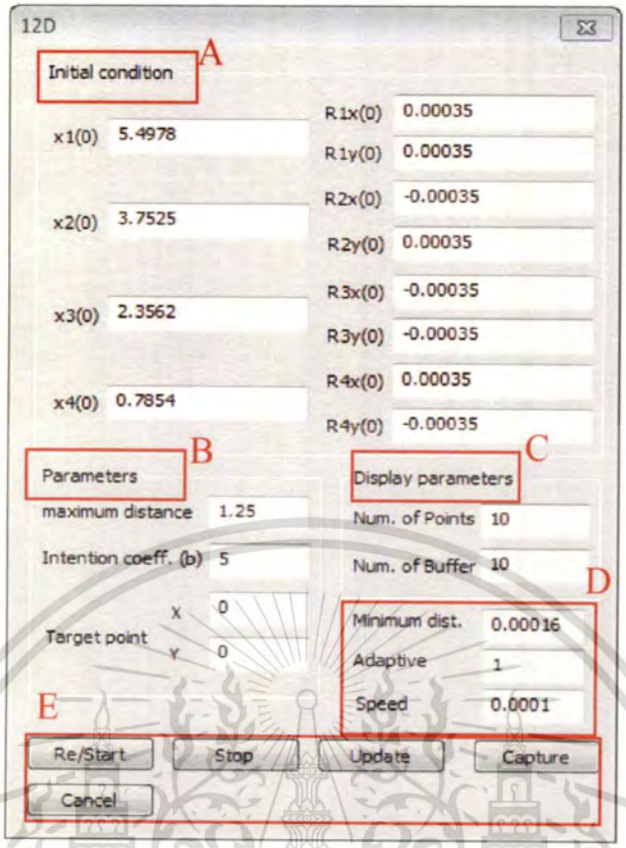


Fig. B.2 The dialog box of the simulator

The following are the description of the main window, which is shown in Fig. B.1.

A. Menus bar—it is only System button used in this simulator. If this button is clicked, then the system dialog box will open enabled us to interfacing with this GUI simulator.

B. Workspace—it is a square field for displaying the robotic network. This workspace is represented in a Cartesian coordinate system. As we can see, there are four dots in different colors represented four mobile robots interacting to each other inside the field. Each dot is followed by tail in smaller dots represented the trajectory. The length of the trajectory can be adjusted from the dialog box. Those trajectories can also be saved in text files.

C. Unit of axes—as the workspace is represented in a Cartesian coordinate system, and then the unit is defined in meter (m). There are four indicators placed in each axis.

D. Counter—indicate the number of iteration that has been done.

E. Time—display the iteration in time (second).

F. Number of Collision—indicate the number of collisions occurred during the simulation.

G. Max. Dist.—inform a max distance between the outer robot and the center of the robotic network.

Figure B.2 shows the dialog box of the non-embedded complex robotic network. This dialog box will appear if the System button in the main window is pressed. The dialog box provides the user for interacting with the system under simulation. We can configure all parameters such as initial conditions, initial positions, interaction coefficients, velocity of robots, and so on. The following summarize the dialog box.

A. Initial condition—those forms are provided for setting initial conditions and position of the robots in the field of workspace. The left side consists of four initial conditions corresponding to s_i in Eq. (4.6) and the right side consists of four pairs of robot coordinates corresponding to the point (x_i,y_i) in Eq. (4.6) on the Cartesian system.

B. Parameters—consist of four forms but only two forms used in the simulation. Maximum distance defines the length of the side of workspace. Intention coefficient is corresponding to parameter w in Eq. (4,6). A target point is not defined yet in this simulation.

C. Display parameters—consist of two forms, firstly, the number of points defined the length of trajectory which is displayed on the main window. Secondly, the number of buffer allocates the memory to keep the trajectory being displayed on the main window.

D. Minimum dist.—defines the minimum distance allowed between two robots. Adaptive form is not defined yet in this simulation. Speed defines the velocity of the robot in *m/s*.

E. Control buttons—consists of five buttons to control the simulation.

1) Re/Start is the start or the restart button. If this button is pressed, then the simulation will start from the beginning, and all configurations will be restarted.

2) Stop button has a function to stop the running simulation.

3) Update button will update the parameters with a new one except the initial conditions and positions. So, only Parameters in B, C, and D will be able to change.

4) Capture button will save all trajectories in separated files for each robot in accordance with the number of points set in Display parameters C. The files will be named based on the time is captured.

5) Cancel button is a simple exit button for terminating the running simulation.

Appendix C

The Downstream Impact of Complex Robotic Network to Science Education

In the view of upstream impact, this research mentioned in the previous chapters have been contributed the knowledge in the field of complex network. Viewing in the opposite direction, this research also contributes to the downstream for science education. Complex robotic network explains how the simple law plays the important role in the behavior of the networks. In the science education point of view, the complex robotic network provides a media for understanding many natural and technological phenomena. The analogies spread widely from the general phenomena that exist surrounding us till the implementation of a specific application. For example, we are a unit of the social complex network comprised humans interaction with each other emerging a complex behavior in daily activities. To demonstrate the fascinating phenomena, we design a robotic network searching agents for science education.

For downstream impact, it was our pleasure to welcome students from the Senior High School of Suankularbwittayalai of Nontha Buri Thailand on June 22, 2011 and Bunhran-Jamsai Polytechnic Collage Suphan Buri Thailand on June 29, 2011. Two months later on August 30, 2011, the former and the new President of Japan International Cooperation Agency (JICA) – Japan also visited us and it was our great opportunity to show the research work in the complex robotic network. The following pictures depict their visit.

C1. Students from Senior High School of Suankularbwittayalai of Nontha Buri Thailand

Fig. C.1 The author (blue shirt) explained the structure of robot to students and supervised by the author's advisor (long shirt with glasses)

Fig. C.2 The author explained the function of overhead camera in the experiment

Fig. C.3 The author answered questions from the students

C2. Students from Bunhran-Jamsai Polytechnic Collage, Suphan Buri, Thailand

Fig. C.4 Authors explained the structure of robots and the motivation of the research experiment to the students and the teacher

Fig. C.5 Students were enjoy the demonstration of the non-embedded complex robotic network

Fig. C.6 The author answered questions from the students

C3. The Former and the New Presidents with Staffs of JICA/Aun-SeedNet on August 30, 2011

Fig. C.7 Welcome speech from Assoc. Prof. Dr. Pitikhate Sooraksa (the most right) for the former and the new President of JICA AUN/SEEd-Net (the third from the left) and the staffs

Fig. C.8 The author explained the non-embedded complex robotic network

Fig. C.9 Assoc. Prof. Dr. Pitikhate Sooraksa explained the results of the experiment

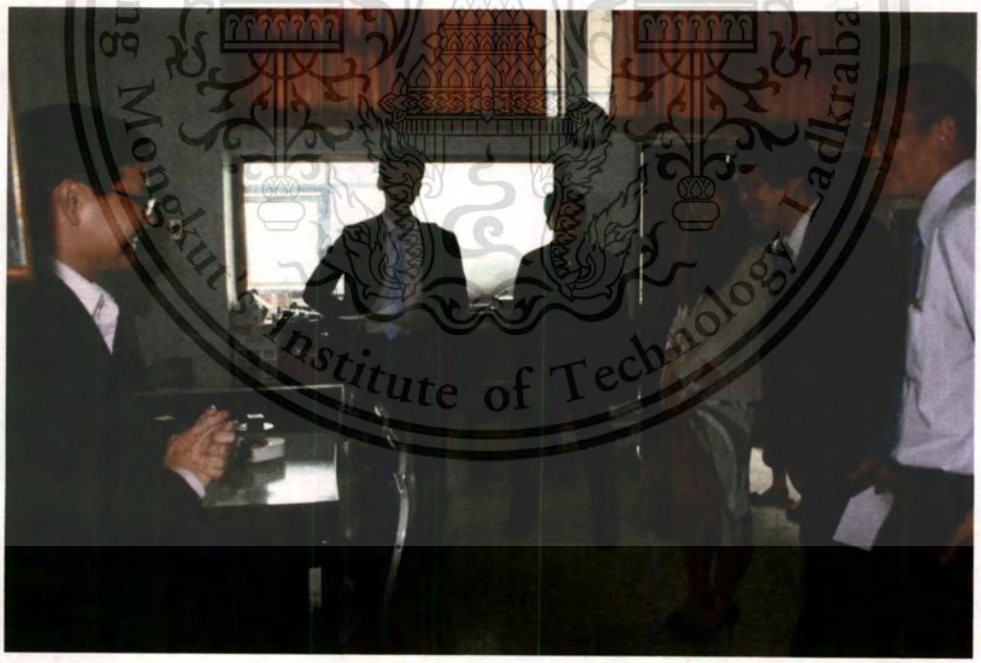


Fig. C.10 The new president of JICA AUN/SEEd-Net gave a good impression and complimented to this research work

Related Publications

1. **Nur Rohman Rosyid** and Pitikhate Sooraksa [2009] "Minimum Complex Network of Chaotic Robots," *in Proc. Of Asia Simulation Conference, JSST-2009*, p.ID040.
2. **Nur Rohman Rosyid**, Masayuki Ohrui, Hiroaki Kikuchi, Pitikhate Sooraksa and Masato Terada [2010] "A Discovery of Sequential Attack Patterns of Malware in Botnets," *In Proc. Int. Conf. IEEE System, Man, and Cybernetic (SMC)*, Istanbul-Turkey, pp. 2564-70.
3. **Nur Rohman Rosyid**, Masayuki Ohrui, Hiroaki Kikuchi, Pitikhate Sooraksa and Masato Terada [2011] "Analysis on the Sequential Behavior of Malware Attacks," Vol.E94D, No.11, pp. 2139-2149.
4. **Nur Rohman Rosyid and** Pitikhate Sooraksa **[2012]** "Experiment on Complex Robotic Networks," *In Proc. Int. Symposium on Technology and Sustainability*, p. CIT009.

Author Biography

Nur Rohman Rosyid was born in Indonesia. He received B.Eng and M.Eng degree, both in Electrical Engineering from Gadjah Mada University, Yogyakarta, Indonesia 2002 and 2005, respectively. In 2002, he joined University of Muhammadiyah Purwokerto as a lecturer until 2005, then he moved to Diploma Program of Electrical Engineering, Gadjah Mada University as a lecturer.

His research interests include chaotic system, complex network and network security. He received a scholarship from JICA / AUN SEED-Net for achieving his Doctoral degree at the School of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang (KMTL), Bangkok, Thailand.