

Detection in Distributed Sensor Networks

A Thesis

Submitted to the Faculty

of

Drexel University

by

Erwei Lin

in partial fulfillment of the
requirements for the degree

of

Doctor of Philosophy

November 2005

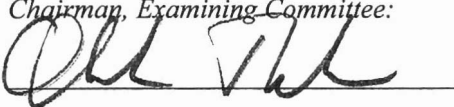
Drexel University
Office of Research and Graduate Studies
Thesis Approval Form
(For Masters and Doctoral Students)

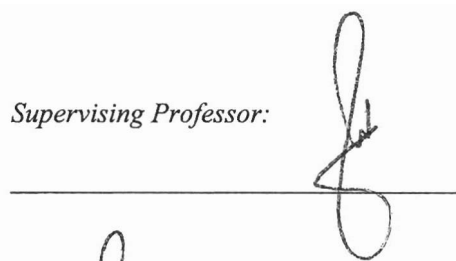
Hagerty Library will bond a copy of this form with each copy of your thesis/dissertation.

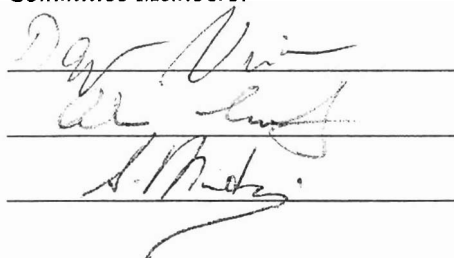
This thesis, entitled Detection in Distributed Sensor Networks

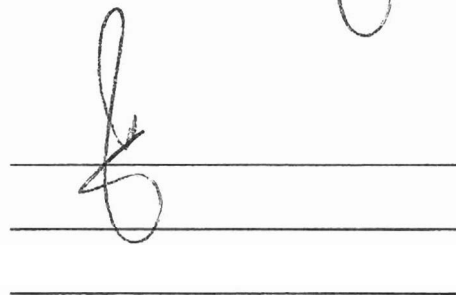
and authored by Erwei Lin, is hereby accepted and approved.

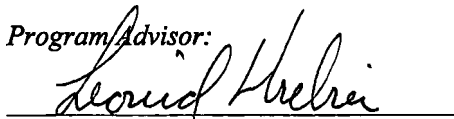
Signatures:

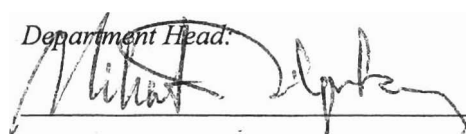
Chairman, Examining Committee:


Supervising Professor:


Committee Members:




Program Advisor:


Department Head:
 11/17/05

Acknowledgments

It would be selfish to say that this dissertation has been the product of my efforts only. Many people have contributed to its completion in many ways: scientific, technical, spiritual, psychological. Some of those I would like to thank here.

First of all I would like to express my deepest gratitude to my advisor, Professor Moshe Kam, who, over the past six years, has not only been teaching me the way of devoted, professional work ethics, but also always standing by my side, offering support and encouragement when I have difficult time. Dr. Kam has been a very dear friend without the unconditional support of whom I would not have accomplished the things I have accomplished up to now.

I am very grateful to Dr. Leonid Hrebien, who, along with Dr. Kam, has provided me a chance to apply my knowledge to physiological signal processing. The research has greatly broaden my knowledge and enriched my life.

Our study was supported by the Office of Naval Research (ONR) through award number N00014-02-1-0787 under the development of "Toward Development of a Virtual Distributed Control System" project. In particular I want to thank Dr Andres Lebaudy and YiFeng Xu from Fairmount Automation, Inc. for their support and appreciation.

Many thanks go to the members of my thesis committee, Dr. Andres Lebaudy, Dr. Spiros Mancoridis, Dr. Dagmar Niebur, and Dr. Oleh Tretiak for their careful review on the thesis and their valuable comments.

I thank my wife, Ming, for always believing in me and consistent supporting me in all circumstances spiritually, emotionally. My world will not be the same without her in it.

Nothing can really express my feelings and gratitude towards my parents ShuSen and QiongJun for the values they have given me, who brought me to this wonderful world,

educated me, and offer me a beautiful life.

My appreciation also goes to my colleagues at the Data Fusion Lab (DFL). Special thanks go to Li Bai , Yingqin Yuan, Lit-Hsin Loo, Ramasubramaniam Achuthanan, Edward Woertz, Mianyu Wang... They have been very helpful in many aspects of both my research and life.

Table of Contents

List of Tables	viii
List of Figures	ix
Abstract	xii
Chapter 1. Introduction	1
1.1 Distributed Sensor Networks	1
1.2 Distributed Detection in Sensor Networks	2
1.3 Thesis Organization	4
1.4 Main Achievements of This Study	5
Chapter 2. Detection in Distributed Sensor Network Using Single Random Access Channel	7
2.1 Introduction	7
2.2 Parallel Decision Fusion Using a Shared Communication Channel	9
2.3 The Local Sensor Model	11
2.4 DFC Decision Schemes	20
2.4.1 Full state is available: decision rule based on the MAP estimator	20
2.4.2 Partial state availability: decision rule based on the EM estimator	23
2.5 Simulation Results	26
2.6 Conclusion and Discussion	29
Chapter 3. Distributed Detection in LonWorks Application	38
3.1 Introduction	38
3.2 An Overview of the Basic Mechanism of the LonWorks Protocol	39

3.2.1	The predictive p -persistent CSMA algorithm implementation in LonTalk	39
3.2.2	Message services	41
3.3	Modeling of the LonWorks Communication Protocol	41
3.3.1	Model for p -persistent CSMA (Unacknowledged service in LonTalk protocol)	43
3.3.2	Model for predictive p -persistent CSMA (Acknowledged service in LonTalk protocol)	43
3.4	Integrated Simulation Model	49
3.4.1	Design concepts	49
3.4.2	Supported functionalities	49
3.4.3	Node model	50
3.5	Model Validation	56
3.5.1	Validation of collision probability	56
3.5.2	Validation of channel throughput and bandwidth utilization	59
3.6	Conclusion and Discussion	61
Chapter 4. An Efficient Method for DOA Estimation: Unitary Improved Polynomial Rooting		62
4.1	Introduction	62
4.2	Model Definition and Problem Formulation	63
4.3	Improved Polynomial Rooting	66
4.4	Proposed Method– Unitary-IPR	67
4.5	Simulation Results	71
4.5.1	Comparisons of RMSE and FPOC vs SNR	72
4.5.2	Comparisons of RMSE and FPOC vs number of sensor	73
4.5.3	Comparisons of RMSE vs number of snapshots	74

4.6	Conclusion	75
Chapter 5. Conclusion and Future Research		77
5.1	Technical Summary	77
5.2	Future Research	77
Appendix A. LonWorks® Analysis Toolkit OPNET Model User's Guide		79
A.1	Overview	79
A.1.1	Introduction of LonWorks OPNET Model	79
A.1.2	Features supported with this release (notation follows standard ANSI/EIA-709.1-A)	79
A.1.3	Audience	81
A.1.4	Content	81
A.1.5	Related Documentation	82
A.2	Getting Started	83
A.2.1	System Requirement	83
A.2.2	Installation	83
A.2.3	Contact information	84
A.3	General Overview	84
A.3.1	Components of the Model	84
A.3.2	Model Parameters	87
A.3.3	Packet Format	88
A.3.4	Collected Statistics	89
A.3.5	Features supported with this release (notation follows standard ANSI/EIA-709.1-A)	89
A.4	Example	89
A.4.1	Creating the Network Model	91

A.4.2	Collecting Statistics	95
A.4.3	Executing the Simulation	96
A.4.4	Analyzing the simulation results	99
	Bibliography	109
	Vita	115

List of Tables

2.1	The set of parameters used in simulation	30
3.1	Summary of supported functionalities	51
A.1	Features supported with this release	80
A.2	Statistics stored in vector output file	90
A.3	Statistics stored in scalar output file for LonWorks node device	90
A.4	Initial settings of sample scenario	92
A.5	Initial settings of sample scenario	99

List of Figures

1.1	Parallel decision fusion structure	3
1.2	Parallel decision fusion with a common LD-DFC channel	4
2.1	Markov chain model for the dynamic of local sensor	12
2.2	Probability of a sensor transmit in a random slot	19
2.3	Comparison of error probability (MAP)	27
2.4	Comparison of error probability (EM)	28
2.5	Comparison of error probability with the CRA proposed in [68]	29
2.6	Probability of transmission of a specified sensor vs. Time under the conditions in Table 2.1	31
2.7	Probability of Successful transmission vs. Time under the conditions in Table 2.1	32
2.8	Probability of Idle state vs. Time under the conditions in Table 2.1	33
2.9	Probability of collision state vs. Time under the conditions in Table 2.1	34
2.10	RMSE of the MAP estimator when channel states are fully available	35
2.11	RMSE of the EM estimator when channel states are not fully available	36
2.12	Receiver operating characteristics for the system. $N = 20$, $P_d = 0.7$, $P_f = 0.3$	37
3.1	Packet cycle of LonWorks protocol	40
3.2	Block diagram of the p -CSMA algorithm	42
3.3	Markov chain model for the backlog window size of LonWorks sensor	44
3.4	Hierarchical structure of network model	50
3.5	Finite state machine of MAC module	53
3.6	Finite state machine of Network and Transaction module	54

3.7	Finite state machine of Transport module	55
3.8	Finite state machine of Application module	56
3.9	Comparison of channel collision probability	58
3.10	Comparison of Node collision probability	59
3.11	Comparison of Channel Throughput	60
3.12	Comparison of Bandwidth Utilization	61
4.1	The RMSE vs. SNR performance of Unitary-IPR compared to MUSIC, root-MUSIC, TLS-ESPRIT, and IPR($q = 3, p = 10$, and 200 snapshots)	72
4.2	The Floating Point Operation Counts vs. SNR performance of Unitary-IPR compared to other methods($q = 3, p = 10$, and 200 snapshots)	73
4.3	Comparison of RMSE vs. Number of Sensors ($q=3, SNR=5dB, 200$ snapshots)	74
4.4	Comparison of FPOC vs. Number of Sensors($q=3, SNR=5dB, 200$ snapshots)	75
4.5	Comparison of RMSE vs. Number of Snapshots ($q=3, p=10, SNR=5dB$)	76
A.1	LonWorks Network Analysis Toolkit OPNET module	81
A.2	Adding the LonWorks Model to Modeler	82
A.3	LonWorks Object Palette	84
A.4	lon_device_module	85
A.5	lon_protocol_analyzer model	86
A.6	TP/FT-10 Channel Model	86
A.7	TP/FT-10 Tap	87
A.8	The various LonWorks Device attributes	87
A.9	Frame format of MAC layer Protocol data unit	88
A.10	Frame format from OPNET	88
A.11	Available statistics of LonWorks OPNET model	91
A.12	Adding the LonWorks Model to Modeler	92

A.13 Rapid Configuration	94
A.14 Network Created by Rapid Configuration	95
A.15 Adding Protocol Analyzer	95
A.16 Node Attributes Dialog Box	96
A.17 Choosing Available Statistics	97
A.18 Setting the Packet Interarrival Time	98
A.19 Simulation Set	99
A.20 Select Scalar Panel Data	100
A.21 Simulation Result 1	101
A.22 Simulation Results 2	102
A.23 Displayed Statistics	103
A.24 Simulation Results 3	103
A.25 Displayed Statistics	104
A.26 Simulation Results 4	105

Abstract
Detection in Distributed Sensor Networks
Erwei Lin
Moshe Kam, Ph.D.

This thesis describes detection and communication algorithms for distributed sensor networks.

In the first part of the thesis, we investigate a new architecture for distributed binary hypothesis detection by employing a Collision Resolution Algorithm (CRA), where all local sensors share a common channel to communicate with the decision fusion center. This architecture is important in the design of sensor fields, where a large number of distributed sensors share a single “emergency” channel.

In the second part of the thesis, we discuss an industrial application of such a distributed detection system, namely, the LonWorks control network. We concentrate on the predictive p -persistent CSMA protocol implemented in the MAC layer of LonWorks protocol, which was proposed by the Echelon Corporation in the 1980s. In order to model this algorithm, we expand the CRA model developed in the first part to analyze variable-length messages. Predictions of the model are compared to an OPNET simulator of LonWorks, and to results from a physical network.

Finally, we propose a direction-of-arrival (DOA) algorithm for sensor networks. It employs an improved polynomial rooting method using unitary transformations.

Chapter 1. Introduction

1.1 Distributed Sensor Networks

A Distributed Sensor Network (DSN) consists of multiple sensor nodes that are capable of communicating with each other and collaborating on a common sensing goal [18, 46, 53]. Interest in DSNs includes modern military applications, which are expected to use a large number of inexpensive sensors distributed from an aerial platform, and then self organize to perform useful detection and estimation tasks.

DSNs are different than other popular distributed networks, which are typically built to transfer data between nodes. In DSNs, nodes often collect information from local sensing resources, and then transmit the combined and processed result to a user. The user is more likely to want information from a region or a set of nodes [39, 40] rather than from individual nodes. In some cases, the user may not even need the detailed information of each individual node and only seeks an aggregated decision or estimate.

It is usually quite expensive to construct a distributed sensor network with nodes communicating through a wired medium. Improvements in wireless communications have reduced the costs of this technology and most sensor networks planned at the present time are wireless [17, 65].

The DSN we assume in this study is a collection of unattended devices that self-organize and have a degree of fault tolerance. It is expected that some nodes may fail, and that some will lose their energy source intermittently or permanently. The DSN must therefore incorporate multiple levels of redundancy. The straightforward solution is to use more sensor nodes than strictly necessary to cover an area. Redundancy in the number of nodes allows for node attrition and increases the reliability of the system [13].

1.2 Distributed Detection in Sensor Networks

Distributed detection in DSN refers to the acquisition, detection, and integration of information gathered by the sensors for the purpose of hypothesis testing. The objective is to provide optimal or near-optimal use of the available information for tasks such as target detection or identification of a threat. Although the original motivation for distributed detection in sensor networks in the early 80s was rooted in military radar applications [24, 30, 38, 66], implementation of multi-sensor fusion systems in civilian applications is now common. In one proposed mode of operation, nodes may lie dormant for long periods waiting for some change (such as the presence of a chemical agent). A forest fire management system is envisaged as an example [8, 14, 42]. For such a system, many thousands of nodes may be preemptively deployed in the area at risk. An increase in temperature or the CO_2 level may trigger local sensors to send warning signals to a user. Many sensors may combine their alarm signals to yield higher-quality information before the user is informed. The objective is to provide aggregated decisions that have simultaneously higher detection rate and lower false alarm rate when compared to individual sensor decision.

There is a 30 year history of research activities in distributed detection for sensor networks. Among these networks, the parallel configuration shown in Figure 1.1 has attracted the most attention. The architecture is comprised of three key components: N local sensors, a decision fusion center (DFC), and a communication channel between each local sensor and the DFC. Typically, the task of such a system is binary hypothesis testing [26, 60, 70]. During the operation, the local sensors observe the phenomenon and make a local decision to determine whether to accept the null hypothesis H_0 ($u_i = 0$, “target absent”) or the alternative hypothesis H_1 ($u_i = 1$, “target present”). It is often assumed that all local observations are statistically independent, conditioned on the hypothesis. The local decision vector $\{u_i\}$, $i = 1, 2, \dots, N$ is then transmitted to the DFC through N dedicated communi-

cation channels, one for each local decision. The DFC then generates the global decision, u_0 from the local decisions.

The Parallel architecture requires the determination of decision rules for both the local (u_1, u_2, \dots, u_N) and global (u_0) decisions, so that the global objective function (*e.g.*, minimum probability of error) can be minimized. The majority of studies of this architecture were devoted to the design of local and global decision rules for Bayesian performance indices and for the Neyman-Pearson (NP) criterion [15, 34, 49, 50, 55–57]. Various extensions of these results, especially for large-scale networks, are summarized in the survey paper [59].

A key assumption of the parallel architecture is the availability of dedicated channels between the local detectors and the DFC. However, in many applications, such as DSNs, such multiple channels are too costly to install and maintain. Instead, sensors use a single over-the-air channel and efficient use of this shared resource become an important objective.

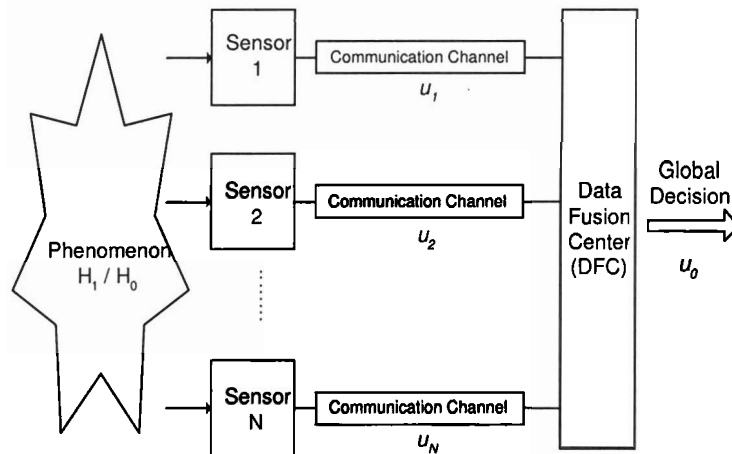


Figure 1.1: Parallel decision fusion structure

Such a DSN is illustrated in Figure 1.2. Here all the local sensors rely on a single, time-slotted, random access channel to send the local decisions to the DFC. Due to the randomness of the access mechanism, during each time slot the channel has three oper-

ational conditions, namely *success* (a single local decision was transmitted successfully), *idle* (none of the local sensors attempted to transmit), and *collision* (more than one local sensor attempted to transmit). In our study, we assume that all the local sensors and the DFC are able to detect the state of channel at each time slot. These schemes are geared towards sensor field applications where alerts are sent over the shared channel only when a threat is detected, and a centralized communication controller is not practical.

Recently, Yuan and Kam [67, 68] have studied the performance of the system shown in Figure 1.2. They analyzed two schemes: one did not use a collision resolution algorithm (CRA) but used instead the statistics of successful transmissions and collisions to discover the observed phenomena. The other used a simple CRA with dynamic retransmission probability. Simulations in [67, 68] showed that the system without CRA would not converge to the optimal performance while the system with CRA appears to converge to the performance of the parallel architecture in Figure 1.1 (which was calculated in [26]).

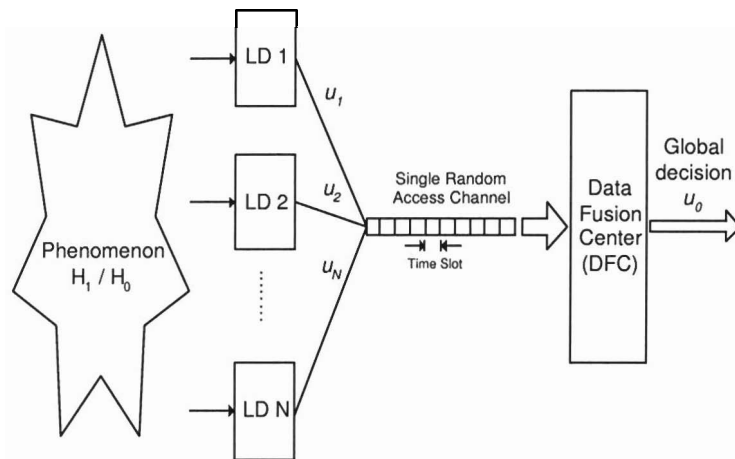


Figure 1.2: Parallel decision fusion with a common LD-DFC channel

1.3 Thesis Organization

This thesis consists of three components. Following this introduction, we study (chapter 2) the decentralized system structure described in Figure 1.2, and propose a more sophis-

licated CRA than the one studied in [68]. We use the results to investigate (in chapter 3), an industrial application of distributed detection, using the LonWorks control technology. We concentrate on the predictive p -persistent CSMA protocol implemented in the MAC layer of the LonWorks protocol, which was proposed by the Echelon Corporation in the 1980s. We expand the CRA of sensor model provided in chapter 2, so it can be used to analyze variable-length messages used by the LonWorks architecture. In chapter 4, we study another application of sensor networks, direction-of-arrival(DOA) of a target. We use a variant of the improved polynomial rooting method to provided the DSN with DOA capability¹.

1.4 Main Achievements of This Study

In this study, We expand on Yuan and Kam's studies [67, 68] of the DSNs that share communication channels by providing a more sophisticated CRA to the local sensor, namely the predictive p -CSMA mechanism. This primary advantage over the Yuan-Kam CRA is that implementation requires less synchronization efforts (see Section 2.1). We further develop decision schemes for such distributed detection systems under two different channel conditions, namely, (1) the full channel state is available to the DFC; and (2) only partial channel state information (successful transmissions) is available to the DFC.

We also discuss an industrial application of such DSNs, namely, the LonWorks control network. We study the predictive p -persist carrier sense multiple access (CSMA) algorithm using an analytical model, simulations, and physical experiments. A analytical model for the algorithm is developed based on the Markov-chain model we provide in Chapter 2, as well as a OPNET-based simulation of the model. Predictions of the analytical model and simulation are compared to experimental data collected from a network of six SMART I/O

¹Admittedly the locations of sensors for this application require higher regularity than the sensor networks envisioned for detection in sentry or chemical/biological threat applications.

ADRII2-F units connected through TP/FT-10 channel.

Finally, we proposed a variation of the improved polynomial rooting (IPR) method for DOA estimation of multiple targets by a sensor network. The variation, unitary IPR (UIPR), transforms the complex-valued covariance matrix of the sensor signals to a real-valued matrix using unitary transformations. Then the IPR method is applied to determine the DOA of the targets. Simulation results indicate the potential improvement provided by our approach compared with MUSIC, Root-MUSIC, ESPRIT, and IPR.

Chapter 2. Detection in Distributed Sensor Network Using Single Random Access Channel

2.1 Introduction

Interest in distributed detection, and the fusion of decisions from decentralized sensors, was spurred by real-world problems, many of which are related to military surveillance applications. Distributed detection has been widely studied, and many architectures and performance results are available [7, 15, 27, 60–63, 69, 71, 72]. From the viewpoint of performance, it is desired for the local detectors in a multiple sensor system to send the raw sensor data to a fusion center where optimal integration algorithm will be employed. Such an approach has the potential to yield optimal detection performance as there is small or no information loss in the communication process. However, in many practical situations, communicating the raw data to the the fusion center may become very expensive and be limited by physical constraints (*e.g.*, communication bandwidth). These limitations often require that the observation at local sensors be compressed before transmission to the fusion center.

Distributed sensor networks (DSNs) are an emerging technology for monitoring a volume of surveillance with a densely distributed network of wireless sensors. Each sensor has limited communication and computation ability and can sense the environment in one or more modalities, such as acoustic, seismic, and infrared. A wide variety of applications are being envisioned for sensor networks, including disaster relief [37], border monitoring [43], condition-based machine monitoring [58], and surveillance in battlefields [20].

Future DSNs are expected to employ a large number of inexpensive sensors whose resolution, bandwidth, and power are limited. Moreover, the communication channels between the local sensors and the central processing center (also known as the fusion center) are expected to be bandwidth-constrained. In such circumstances, the local sensor readings are

often compressed, in the extreme into 1 bit decisions (target present/absent). When these compressed readings are collected, an elaborate processing mechanism may be needed to develop an estimate of the original data that were observed.

Rago *et al.* [48] studied the decentralized detection problem with communication constraints, where the sensors employ a “send/no send” strategy to reduce the communication requirements from the communication channel between sensors and the fusion center. In an earlier study, Longo *et al.* [36] considered the bandwidth constrained problem using an information-theoretic framework.

The motivation for our study is the additional constraints on many sensor networks that limits the sensor communications to a single shared over-the-air channel (figure 1.2).

Yuan and Kam [68] have recently studied this restriction, and examined how the Data Fusion Center (DFC) in Figure 1.2 performs (i) without implementing a collision resolution algorithm (CRA), and (ii) with a simple CRA similar to slotted-ALOHA (and with dynamically updated retransmission probability). The basic idea is to divide the time into non-overlapping contention windows, and to have all the sensors that detected a target (those with $u_k = 1$ in our notation) attempt to inform the DFC during the contention window. In [68] the contention window is divided into time slots. All sensors that detected a target and were not able to transmit the decision successfully attempt to transmit this decision with a certain probability during the next time slot. The performance of the system with CRA appeared to have converge to the optimal performance. The major limitation of [68] is that all local sensors have to maintain synchronization of the contention window, since the transmission probability is reset at the beginning of the contention window. Moreover, during the operation, all local sensors have to calculate and maintain the *same* transmission probability. Deviation from the shared probability by some local sensors would cause significant deterioration in overall performance.

In our distributed detection architecture, all local sensors are connected to the shared communication channel. As in [67, 68], only the sensors that detect the presence of a “tar-

get" ($u_k = 1$) will plan to transmit its 1-bit decision to the DFC. In order to maximize the probability of successful transmission, a CRA mechanism, similar to the p -CSMA algorithm implemented in IEEE 802.11 WLAN protocol [11], is employed by every local sensor. It causes the sensors to update their transmission probability every time they transmit. The DFC studies the output of the communication channel and makes a decision about target presence/absence based on the information gathered over a fixed time interval immediately preceding the present time. At each time slot, the communication channel is in one of three possible states: *success*, *idle* or *collision*. We assume that at each time slot n sensors attempted a transmission. For a time slot to be in the idle state, $n = 0$; to be in *success* state, $n = 1$; to be in a *collision* state, $n > 1$. Since the DFC produces a new opinion after each time slot, the proposed scheme can be implemented as an on-line, non stop process.

The rest of this chapter is organized as follows. Following the introduction, we describe the parallel decision fusion using a shared communication channel. In section 2.3 we describe the CRA mechanism. We then describe the decision making at the DFC for two conditions: 1) the DFC possesses full state (*i.e.*, the number of *success* slots, *idle* slots, and *collision* during a defined of time interval) and 2) only the number of successful transmissions during a certain interval is available to the DFC. Next, we simulate the system performance and compare performance of the single-channel system (figure 1.2) to the performance of a parallel binary system with dedicated sensor-DFC channels [26].

2.2 Parallel Decision Fusion Using a Shared Communication Channel

We refer to figure 1.2, where the studied architecture is presented. The system is comprised of N local sensors, all of which communicate with the DFC through a shared, time-slotted, single random-access channel. The task is binary hypothesis testing. We denote the two hypotheses as H_0 and H_1 , where H_0 represents the null hypothesis (target absent) and H_1

represents the alternative hypotheses (target present). The *a priori* probabilities of H_0 and H_1 are assumed to be constant and known, and denoted as P_0 and P_1 , respectively.

We use the time slot of the communication channel as the time unit of the system. A discrete, integer time slot scale is thus adopted. One slot is considered long enough for each sensor to sample the environment, make the local decision, and transmit the decision to the DFC if the channel is available and the local sensor “believes” the target is present (“1” is sent to the DFC by the local sensor). The DFC makes a global decision by estimating the number of transmitting sensors during a specified time interval that consists multiple time slots (this number is compared to a threshold to decide whether or not H_1 should be accepted. The performance index is probability of error). We further assume that all sensors are only allowed to transmit at the beginning of each time slot so that the collisions will be experienced immediately by each local sensor during the subsequent time slot. In other words, in our study, all the sensors need to be synchronized for the unit of time instead both of the unit of time and the length of the contention window as required in [67, 68].

Let $u_k^T \in \{0, 1\}$, $k = 1, 2, \dots, N$, $T = 1, 2, \dots$ be the decision made by k th local sensor at time T for the observation of the underlying phenomenon. The decision $u_k = 1$ is used to represent acceptance of H_1 by the k th local sensor, and $u_k = 0$ is used to represent acceptance of H_0 by the k th local sensor. $u_0^T \in \{0, 1\}$ is used to denote the global decision made by the DFC at time T . We assume that all the local sensors employ the same fixed decision rule, and that they all have same false alarm probability $P_f = P(u_k = 1|H_0)$ and missed detection probability $P_m = P(u_k = 0|H_1)$. Temporal and spatial statistical independence are assumed here for all observations of all N sensors, conditioned on the hypothesis.

2.3 The Local Sensor Model

Let $z_k \in R^l$ (for some integer l) be a vector of local observations available to the k th sensor. During any time slot, the sensor uses $z_k \in 0, 1$ in a decision rule $u_k = g(z_k)$ to make decision u_k . After processing the observations z_k locally, if the local sensor detects the target is present ($u_k = 1$), it will try to transmit its decision to the DFC through the single random access channel. The local sensor will use the CRA to schedule a transmission attempt.

Using our CRA, each sensor maintains its local backlog parameter W_i , which depends on the number of successful transmissions and collisions experienced. W_i is used to generate the random waiting period (rwp) that is uniformly selected from the range $(0, 1, \dots, W_i)$. Once a local sensor was able to transmit its decision successfully, it changes its local backlog parameter so that the period of rwp is decreased. If the attempt to transmit was unsuccessful (caused a collision), the local sensor changes its backlog parameter in the direction of increasing rwp . We increase or decrease W_i by a fixed amount W_{base} every time a transmission attempt has failed or has succeeded. W_i is calculated as

$$W_i = W_{base}(1 + i) - 1, i \in (0, m), \quad (2.1)$$

where i is defined as the “backlog stage”. It ranges from 0 to the maximum backlog value m . The value of i moves toward m by one step when the transmission attempt has failed, and it moves toward 0 by one step when the transmission attempt has succeeded. Both m and W_{base} are design parameters. Once the value of rwp was uniformly selected from the range $(0, 1, \dots, W_i)$, a timer is set and a countdown is started. The local sensor transmits the 1-bit message when the time reaches zero. The data fusion center determines the global decision u_0 based on the observation of the channel states during a certain fixed interval.

Let $b(t)$ be the stochastic process that represents the rwp counter for a specific local sensor, and $s(t)$ the stochastic process representing the backlog stage $(0, \dots, m)$ of the

sensor at time t . Process $b(t)$ represents the number of remaining slots in rw_p before the sensor starts to transmit its 1-bit decision. As we mentioned in 2.2, a discrete, integer time scale is adopted where t and $t + 1$ correspond to two consecutive time slots, and the rw_p counter of each sensor decreases at the beginning of each time slot. The backlog window size for every sensor depends on the collisions and on the successful reservation attempts experienced by the sensor in the past.

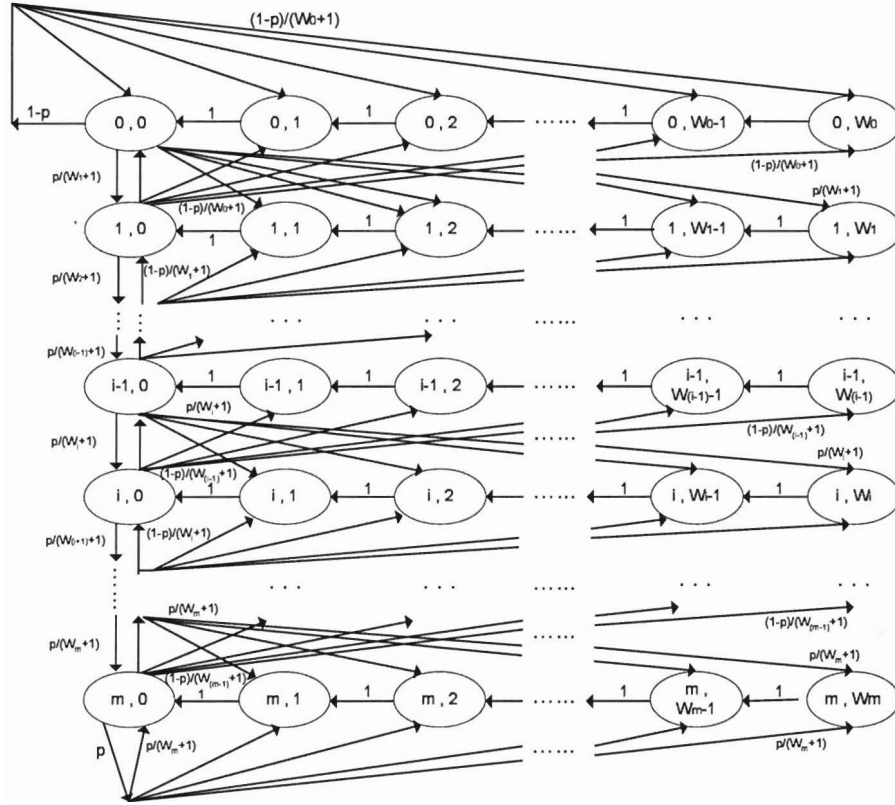


Figure 2.1: Markov chain model for the dynamic of local sensor

The key assumption of our CRA is that a collision occur with the same probability p regardless of the state of the backlog time counter used for this transmission. Based on this assumption, we modified the discrete time Markov chain model in [11], which was originally developed to study the MAC algorithm of IEEE 802.11 protocol for wireless

local area networks. We can model $s(t), b(t)$ by the discrete time Markov chain model presented in Figure 2.1, which is a state transition diagram with m rows, each having $W_i + 1$ ($i = 0, 1, \dots, m$) states. Each state is indexed by two integers $(s(t), b(t))$, so that if $s(t) = s$ and $b(t) = b$, then we are at backlog state s and have b time slots until we will try to transmit. The numbered arrows that connect the states indicate the direction and probability of transition from the source state to the destination state. Transitions between states in the same row occur every time unit. Transitions between rows occur only after a transmission attempt: we move up if the attempt was successful; we move down if the attempt was unsuccessful.

If we adopt the short notation $P\{i, j|m, n\} = P\{s(t+1) = i, b(t+1) = j | s(t) = m, b(t) = n\}$ for the conditional transition probabilities, the one-step transition probabilities can be found as following

Within a row:

$$P\{i, j|i, j+1\} = 1, \quad (2.2)$$

$$i \in (0, m), \quad j \in (0, W_i - 1).$$

Moving to higher row:

$$P\{i, j|i+1, 0\} = \frac{1-p}{W_i+1}, \quad (2.3)$$

$$i \in (0, m-1), \quad j \in (0, W_i).$$

Moving to a lower row:

$$P\{i+1, j|i, 0\} = \frac{p}{W_{i+1}+1}, \quad (2.4)$$

$$i \in (0, m-1), \quad j \in (0, W_{i+1})$$

Staying at the highest row:

$$P\{0, j|0, 0\} = \frac{1-p}{W_0+1}, \quad (2.5)$$

$$j \in (0, W_0).$$

Staying at the lowest row:

$$P\{m, j|m, 0\} = \frac{p}{W_m + 1}, \quad (2.6)$$

$$j \in (0, W_m).$$

Equation 2.2 states that after the sensor generates the *rwp* timer according to the estimated backlog window size, it keeps decreasing until it reaches the zero state of the current layer ($b(t) = 0$).

Equation 2.3 states that if the transmission of the packet at time i is successful (with probability $1 - p$), the sensor moves one stage toward stage 0 and the estimated backlog window size is decreased by 1. The *rwp* for the transmission of the next packet will be generated uniformly from a smaller range $(0, W_i)$.

Equation 2.4 shows that if the transmission of the packet has failed, which means that a collision has occurred with probability p , the sensor moves one step toward stage m and the estimated backlog window size is increased by 1. The new *rwp* will be generated from a larger range $(0, W_{i+1})$.

Equation 2.5 considers the special condition of the sensor being at stage 0 when it transmits a packet successfully.

Equation 2.6 is the special case when the sensor is at the last stage and it fails to transmit a packet.

We designate $b_{i,j} = \lim_{t \rightarrow \infty} P\{s(t) = i, b(t) = j\}$, $i \in (0, m)$, $j \in (0, W_i)$ to be the stationary behavior after the process has been running for a long time. For state $b_{0,0}$, we have the balance equation:

$$(1 - p)b_{0,0} + pb_{0,0} = (1 - p)b_{0,0} + (1 - p)b_{1,0}, \quad (2.7)$$

which gives

$$b_{1,0} = \frac{p}{1 - p} b_{0,0}. \quad (2.8)$$

Similarly, to state $b_{1,0}$, we can establish the balance equation:

$$(1-p)b_{1,0} + pb_{1,0} = (1-p)b_{2,0} + pb_{0,0}, \quad (2.9)$$

substituting $b_{1,0}$ from Equation 2.8 in Equation 2.9, we have

$$b_{2,0} = \left(\frac{p}{1-p}\right)^2 b_{0,0}. \quad (2.10)$$

Using the method of induction, it can be shown that

$$b_{i,0} = \left(\frac{p}{1-p}\right)^i b_{0,0}, \quad i \in (0, m). \quad (2.11)$$

The transition probability of the initial stage $b_{0,j}$, $j \in (0, W_0)$ in the Markov chain model satisfies

$$b_{0,j} = \frac{W_0 + 1 - j}{W_0 + 1} (b_{1,0}(1-p) + b_{0,0}(1-p)). \quad (2.12)$$

The transition probabilities of the last stage $b_{m,j}$, $j \in (0, W_m)$ satisfy

$$b_{m,j} = \frac{W_m + 1 - j}{W_m + 1} (b_{m-1,0}p + b_{m,0}p). \quad (2.13)$$

For the rest of the stages, we have the transition probabilities $b_{i,j}$

$$b_{i,j} = \frac{W_i + 1 - j}{W_i + 1} (b_{i+1,0}(1-p) + b_{i-1,0}p), \quad i \in (1, m-1) \quad (2.14)$$

By substituting Equation 2.8 to Equation 2.12, we get

$$b_{0,j} = \frac{W_0 + 1 - j}{W_0 + 1} b_{0,0}. \quad (2.15)$$

Substitute Equation 2.11 into Equations 2.13 and 2.14

$$b_{i,j} = \frac{W_i + 1 - j}{W_i + 1} b_{i,0}, \quad i \in (1, m-1), \quad (2.16)$$

and

$$b_{m,j} = \frac{W_m + 1 - j}{W_m + 1} b_{m,0}. \quad (2.17)$$

Equations 2.15, 2.16, and 2.17 can be summarized as one equation:

$$b_{i,j} = \frac{W_i + 1 - j}{W_i + 1} b_{i,0}, \quad i \in (0, m), \quad j \in (0, W_i), \quad (2.18)$$

where

$$b_{i,0} = \left(\frac{p}{1-p}\right)^i b_{0,0}. \quad (2.19)$$

Since the sum of the probabilities of all the states in a Markov Chain model is 1, we have,

$$\sum_{i=0}^m \sum_{j=0}^{W_i} b_{i,j} = 1 \quad (2.20)$$

By substituting Equations 2.18 and 2.19, Equation 2.20 can be simplified:

$$\sum_{i=0}^m b_{i,0} \sum_{j=0}^{W_i} \frac{W_i + 1 - j}{W_i + 1} = 1 \quad (2.21)$$

$$\sum_{i=0}^m b_{i,0} \frac{W_i + 2}{2} = 1 \quad (2.22)$$

$$\frac{b_{0,0}}{2} \sum_{i=0}^m \left(\frac{p}{1-p}\right)^i \frac{W_i + 2}{2} = 1 \quad (2.23)$$

From which we can further derive (by substituting Equation 2.18 and $W_i = W_{base}(1 + i) - 1$ into Equation 2.20):

$$b_{0,0} = \frac{2}{(W_{base} + 1) \sum_{i=0}^m \left(\frac{p}{1-p}\right)^i + W_{base} \sum_{i=0}^m i \left(\frac{p}{1-p}\right)^i} \quad (2.24)$$

We can now express the probability τ that a sensor transmits in a randomly chosen time slot. As the transmission occurs only when the backlog counter is equal to zero, regardless

of the backlog stage, it is

$$\tau = \sum_{i=0}^m b_{i,0} \quad (2.25)$$

$$= \sum_{i=0}^m \left(\frac{p}{1-p}\right)^i b_{0,0} \quad (2.26)$$

$$= b_{0,0} \frac{(1-p)^{m+1} - p^{m+1}}{(1-2p)(1-p)^m} \quad (2.27)$$

After substituting and applying the equation of Arithmetic-Geometric series, τ can be expressed as

$$\tau = \frac{2}{(W_{base} + 1) + \frac{W_{base}(p(1-p)^{m+1} + (2m+1)p^{m+2} - (m+1)p^{m+1})}{((1-p)^{m+1} - p^{m+1})(1-2p)}}, \quad p \neq \frac{1}{2}. \quad (2.28)$$

When $p = \frac{1}{2}$, from Equation 2.19 we have $b_{i,0} = b_{0,0}$, substituting this in Equation 2.22 gives us

$$\sum_{i=0}^m b_{0,0} \frac{W_i + 2}{2} = 1 \quad (2.29)$$

$$b_{0,0} = \frac{1}{(W_{base} + 1)(m + 1) + W_{base} \sum_{i=0}^m i} \quad (2.30)$$

$$b_{0,0} = \frac{2}{(m + 1)(W_{base}(1 + \frac{m}{2}) + 1)}, \quad (2.31)$$

substituting it in Equation 2.26, we have

$$\tau = \frac{2}{(W_{base}(1 + \frac{m}{2}) + 1)}, \quad p = \frac{1}{2}. \quad (2.32)$$

Equations 2.28 and 2.32 can be combined as

$$\tau = \begin{cases} \frac{2}{(W_{base}(1 + \frac{m}{2}) + 1)}, & p = \frac{1}{2}; \\ \frac{2}{(W_{base} + 1) + \frac{W_{base}(p(1-p)^{m+1} + (2m+1)p^{m+2} - (m+1)p^{m+1})}{((1-p)^{m+1} - p^{m+1})(1-2p)}}, & \text{otherwise} \end{cases} \quad (2.33)$$

Probability τ is a function of collision probability p which is still unknown. The probability p that a contention attempt collides is the probability that at least one of the remaining

$n - 1$ local sensors transmits in the same time slot. Hence, we are able to get another equation for p and τ :

$$p = 1 - (1 - \tau)^{n-1} \quad (2.34)$$

Equations 2.33 and 2.34 form a nonlinear equation set with unknown p and τ . The unique solution can be found by employing numerical methods that evaluate p and τ for the combinations of W and m . The following steps prove that a unique solution for the equation set exists:

We rewrite Equation 2.34 as

$$\tau = 1 - (1 - p)^{\frac{1}{n-1}} \quad (2.35)$$

τ is a continuous and monotonically increasing variable with respect to the $p \in (0, 1)$ (see Figure 2.2) since $(1 - p)^{\frac{1}{n-1}}$ is a continuous and monotonically decreasing with respect to the $p \in (0, 1)$ when $n \neq 1$. τ increases from 0 ($p = 0$) to 1 ($p = 1$). The τ in Equation 2.33 is continuous and monotonically decreasing in the same range $(0, 1)$. It decreases from $\frac{2}{W_{base} + 1} (> 0)$ to $\frac{2}{W_{base}(m + 1) + 1} (< 1)$. Hence, a unique solution can be always found due to the existence of a single intersection point.

Given that there are n sensors attempting to transmit local decisions to the DFC, the probability that at least one sensor attempts to transmit in any given time slot is given by

$$P_{tr} = 1 - (1 - \tau)^n. \quad (2.36)$$

The probability that an occurring transmission is successful, is given by the probability that one sensor attempts to transmit and the remaining $n - 1$ sensors remain *idle*, provided that at least one transmission occurs in the channel

$$P_s = \frac{n\tau(1 - \tau)^{n-1}}{P_{tr}} \quad (2.37)$$

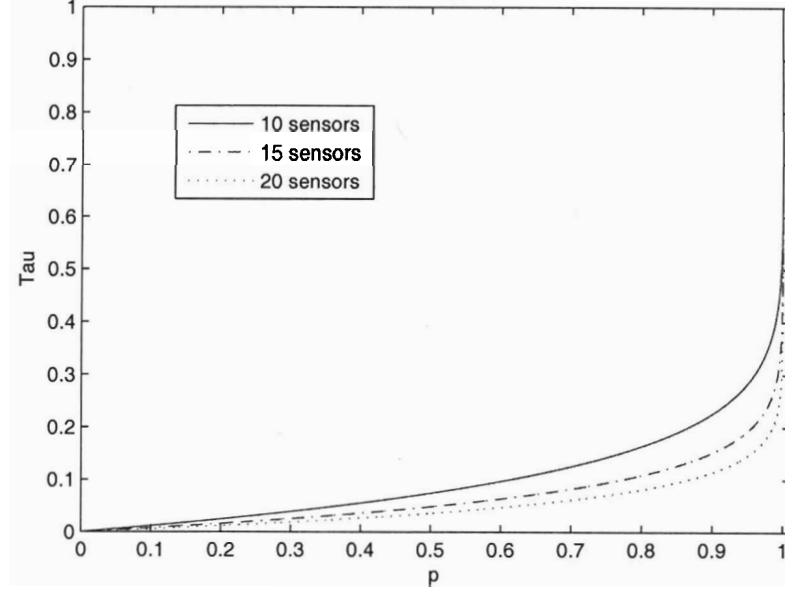


Figure 2.2: Probability of a sensor transmit in a random slot

The probabilities that a randomly selected time slot will be a *success*, *idle* or *collision* ($P_{S|n}$, $P_{I|n}$, and $P_{C|n}$, respectively) are

$$P_{S|n} = P_{tr}P_s \quad (2.38)$$

$$P_{I|n} = 1 - P_{tr} \quad (2.39)$$

$$P_{C|n} = P_{tr}(1 - P_s) \quad (2.40)$$

Let the number of success slots, idle slots, and collision slots during a W window be N_S , N_I , N_C , respectively. Given that there are n sensors attempt to transmit, the probability that $N_S = n_S$, $N_I = n_I$, and $N_C = n_C$ is a trinomial distribution function:

$$P_{n_S, n_I, n_C|n} = \binom{W}{n_S} \binom{W - n_S}{n_I} P_{S|n}^{n_S} P_{I|n}^{n_I} P_{C|n}^{n_C}, \quad (2.41)$$

where $n_S + n_I + n_C = W$, and $\binom{W}{n_S} = \frac{W!}{n_S!(W - n_S)!}$.

2.4 DFC Decision Schemes

Once the local sensors detect the presence of the target, they start to access the channel using the CRA we described in Section 2.3. The DFC makes the decision about the two hypotheses according to the available statistics of the channel states within a W window. In our studies, the size of W is not necessarily needed to be greater than the number of sensors.

We used two estimators in two different conditions to estimate the number of sensors attempting to transmit on the channel during the period of W : 1) Bayesian maximum a-posteriori (MAP) estimator under the condition that the channel state is fully available to the DFC (namely, S for *success* state, I for *idle* state, C for *collision* state), 2) Expectation Maximization (EM) estimator under the condition that only a partial channel state is available to the DFC (only the the number of successful transmission).

2.4.1 Full state is available: decision rule based on the MAP estimator

We assume that the DFC has full awareness of the statistics of the channel states during any time. Let X_l , $l = 1, 2, \dots, W$ be the random state of the l th time slot in a W window, with the realization $x_l \in \{S, I, C\}$, and let $x_{l:m} = (x_l, x_{l+1}, \dots, x_m)$ be the sequence of realization states from the l th to m th time slots, l and m hold the relation $1 \leq l \leq m \leq W$. The MAP estimator calculates the estimate of the total number of transmitting sensors (\hat{N}) using the criterion

$$C_{MAP} = \arg \max_{n \in \{1, \dots, N\}} P(\hat{N} = n | x_{1:k}) \quad (2.42)$$

where n is selected from $\{0, 1, \dots, N\}$. $P(\hat{N} = n | x_{1:k})$ is the probability that there are n sensors attempting to access the channel given the sequence of realization states from the first to k th slot.

$$\hat{n}_k = \arg \max_n P(\hat{N} = n | x_{1:k}) \quad (2.43)$$

$$= \arg \max_n P(x_{1:k} | \hat{N} = n) P(\hat{N} = n) \quad (2.44)$$

The term $P(x_{1:k} | \hat{N} = n)$ denotes the probability that the realization of the channel status is $x_{1:k}$, given that n sensors accepted H_1 during the duration of a W window. For $k \geq 2$ we have

$$\begin{aligned} P(x_{1:k} | \hat{N} = n) &= P(x_{1:k-1}, x_k | \hat{N} = n) \\ &= P(x_k | x_{1:k-1}, \hat{N} = n) \\ &\quad * P(x_{1:k-1} | \hat{N} = n), \end{aligned} \quad (2.45)$$

where

$$P(x_k | x_{1:k-1}, \hat{N} = n) = \begin{cases} P_{S|n}, & \text{if } x_k = S; \\ P_{I|n}, & \text{if } x_k = I; \\ P_{C|n}, & \text{if } x_k = C. \end{cases} \quad (2.46)$$

It is insightful to consider the limited case:

$$P(x_{1:1} | \hat{N} = n) = P(x_1 | \hat{N} = n) \quad (2.47)$$

$$= \frac{P(x_1, \hat{N} = n)}{P(\hat{N} = n)} \quad (2.48)$$

$$= \frac{P(x_1, \hat{N} = n)}{P(\hat{N} = n | H_0) P_0 + P(\hat{N} = n | H_1) P_1} \quad (2.49)$$

$$= \begin{cases} \frac{P_{S|n}}{N(P_0 P_f + P_1 P_d)}, & \text{if } x_1 = S; \\ \frac{P_{I|n}}{N(P_0 P_f + P_1 P_d)}, & \text{if } x_1 = I; \\ \frac{P_{C|n}}{N(P_0 P_f + P_1 P_d)}, & \text{if } x_1 = C. \end{cases} \quad (2.50)$$

We assume that the objective of the DFC is to minimize the probability of error

$$P_E = P(H_0)P(u_0 = 1|H_0) + P(H_1)P(u_0 = 0|H_1). \quad (2.51)$$

We further assume that all the observations at the local detectors are conditionally independent as well as restricted to be identically distributed, and the probabilities of false alarm ($P_f = P(u_k = 1|H_0)$) and detection ($P_d = P(u_k = 1|H_1)$) at each of the local sensors are the same. We can then apply the “K out of N” optimal decision rule [60] (section 3.4) to make the decision about the two hypotheses

$$\hat{N} \underset{H_0}{\overset{H_1}{>}} \frac{\log\left[\frac{P_0}{P_1}\left(\frac{1-P_f}{1-P_d}\right)^N\right]}{\log\left[\frac{P_d(1-P_f)}{P_f(1-P_d)}\right]} = K^*. \quad (2.52)$$

Thus, the optimum value of K for the “K-out-of-N” fusion rule is given by

$$K_{opt} = \begin{cases} \lceil K^* \rceil, & \text{if } K^* \geq 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2.53)$$

where $\lceil \cdot \rceil$ denotes the standard ceiling function, and it rounds the value of K^* to the nearest target integer. The overall probability of false alarm can be expressed as

$$P_F = P(\hat{N} \geq K^* | H_0) \quad (2.54)$$

$$= \sum_{j=1}^N P(\hat{N} \geq K^* | n = j, H_0) P(n = j | H_0) \quad (2.55)$$

and the probability of missed detection is

$$P_M = P(\hat{N} < K^* | H_1) \quad (2.56)$$

$$= \sum_{j=1}^N P(\hat{N} < K^* | n = j, H_1) P(n = j | H_1) \quad (2.57)$$

2.4.2 Partial state availability: decision rule based on the EM estimator

In the previous section, we assumed that the full state of the channel is available to the DFC. However, in some circumstances, the DFC has knowledge only of successful transmissions (the decisions made by the local sensors) during a specified time interval. We are still interested in estimating the number of sensors that are transmitting local decisions on the communication channel given the incomplete data set of the channel states. To solve this problem, we use the Expectation-Maximization (EM) iterative algorithm [19], which is a broadly applicable statistical technique for handling incomplete data models. At each iteration of the algorithm, two steps are performed: (1) E-Step consisting of projecting an appropriate functional containing the augmented data on the space of the original, incomplete data, and (2) M-Step consisting of maximizing the functional.

We assume the complete data set $\chi = \{x_1, x_2, \dots, x_l\}$ is divided into an observed component χ^o (the incomplete data set) and a missing component χ^m . Similarly, each data vector x_l is divided into (x_l^o, x_l^m) . χ^o is assumed to have a postulated pdf as $f(\chi^o, \kappa)$, where $\kappa = (\kappa_1, \dots, \kappa_d)$ is a vector of unknown parameters that we would like to estimate. We denote the pdf of the random vector corresponding to the complete data set χ as $g_c(\chi, \kappa)$. The log-likelihood for κ , if χ were fully observed, would be

$$\log L_c(\kappa) = \log g_c(\chi, \kappa). \quad (2.58)$$

The incomplete data vector χ^o comes from the “incomplete” sample space Υ^i . Since there is a one-to-one correspondence between the complete sample space Υ^c and the incomplete sample space Υ^i , for $x_l \in \Upsilon^c$, one can uniquely find the one-to-one correspondence $x_l^o = F(x_l)$ in Υ^i . Also, the incomplete pdf could always be found by integrating out the complete pdf,

$$g(\chi^o, \kappa) = \int_{\Upsilon^c(\chi^o)} g_c(\chi, \kappa) d\chi \quad (2.59)$$

where $\Upsilon^c(\chi^o)$ is the subset of Υ^c constrained by the relation $x_l^o = F(x_l)$.

Let $\kappa^{(0)}$ be some initial value for κ . At the k -th step, the EM algorithm performs the following two steps:

- **E-Step.** Calculate

$$Q(\kappa, \kappa^{(k)}) = E_{\kappa^{(k)}}\{\log L_c(\kappa)|\chi^o\} \quad (2.60)$$

- **M-Step.** Choose any value $\kappa^{(k+1)}$ that maximizes $Q(\kappa, \kappa^{(k)})$, i.e.,

$$(\forall \kappa) Q(\kappa^{(k+1)}, \kappa^{(k)}) \geq Q(\kappa, \kappa^{(k)}) \quad (2.61)$$

In our study, the DFC receives n_S local decisions during the W window, the probability density function, given the data is

$$g(\chi^o, P_{S|n}) = \frac{W!}{n_S!n_T!} P_{S|n}^{n_S} (1 - P_{S|n})^{n_T} \quad (2.62)$$

where $n_T = W - n_S$, is the summation of the number of the collision and idle slots.

By estimating the $P_{S|n}$, we will know how many sensors are accessing the channel.

Assume that the original value n_T comprises the counts \bar{n}_I and \bar{n}_C , such that $\bar{n}_I + \bar{n}_C = n_T$. The probability of an idle state is assumed $\alpha P_{S|n}$ and the probability of a collision state is assumed $1 - (\alpha + 1)P_{S|n}$, where $\alpha = \frac{1-\tau}{n_T}$. The "complete data" can be defined as $\chi = (n_S, \bar{n}_I, \bar{n}_C)$, where $n_S + \bar{n}_I + \bar{n}_C = W$.

The probability mass function of incomplete data χ^o is $g(\chi^o, P_{S|n}) = \sum g_c(\chi, P_{S|n})$, where

$$g_c(\chi, P_{S|n}) = c(\chi) P_{S|n}^{n_S} (\alpha P_{S|n})^{\bar{n}_I} \{1 - (\alpha + 1)P_{S|n}\}^{\bar{n}_C} \quad (2.63)$$

where $c(\chi)$ is free of $P_{S|n}$, and the summation is taken over all values of χ for which $\bar{n}_I + \bar{n}_C = n_T$.

The "complete" log likelihood is

$$\begin{aligned}\log L_c(P_{S|n}) &= n_S \log(P_{S|n}) + \bar{n}_I \log(\alpha P_{S|n}) \\ &\quad + \bar{n}_C \log\{1 - (\alpha + 1)P_{S|n}\}\end{aligned}\tag{2.64}$$

Our goal is to find the conditional expectation of $\log L_c(P_{S|n})$ given χ° , using the starting point for $P_{S|n}^{(0)}$,

$$Q(P_{S|n}, P_{S|n}^{(0)}) = E_{P_{S|n}^{(0)}} \{\log L_c(P_{S|n}) | \chi^\circ\}.\tag{2.65}$$

As $\log L_c$ is a linear function in \bar{n}_I and \bar{n}_C , the E-step is done simply by replacing \bar{n}_I and \bar{n}_C by their conditional expectation, given χ° .

Consider N_C to be a random variable corresponding to n_C , it is easy to see that the conditional expectation of N_C given n_T is

$$E_{P_{S|n}^{(0)}}(N_C | n_T) = \frac{n_T [1 - (\alpha + 1)P_{S|n}^{(0)}]}{1 - P_{S|n}^{(0)}} = \bar{n}_C^{(0)}\tag{2.66}$$

Further, $\bar{n}_I^{(0)} = n_T - \bar{n}_C^{(0)}$. This completes the **E-Step** part.

In the **M-Step** part, one chooses $P_{S|n}^{(1)}$ so that $Q(P_{S|n}, P_{S|n}^{(0)})$ is maximized. After replacing \bar{n}_C and \bar{n}_I by their conditional expectation $\bar{n}_C^{(0)}$ and $\bar{n}_I^{(0)}$ in the Q function, the maximum is obtained at

$$P_{S|n}^{(1)} = \frac{\bar{n}_I^{(0)} + n_S}{(\alpha + 1)W}\tag{2.67}$$

Now the **E-** and **M-**steps are alternating. At the iteration k we have

$$P_{S|n}^{(k+1)} = \frac{\bar{n}_I^{(k)} + n_S}{(\alpha + 1)W}\tag{2.68}$$

$$= \frac{n_T - \bar{n}_C^{(k)} + n_S}{(\alpha + 1)W}\tag{2.69}$$

2.5 Simulation Results

The model of a local sensor specified in Section 2.3, together with the two fusion schemes under two different channel conditions allow the development of numerical procedures for the evaluation of our distributed detection system, in terms of probabilities of channel states and detection error.

Figures 2.3 and 2.4 show the global probabilities of error (calculated from Equation 2.51) against the contention window size for the system with $P_0 = P_1 = 0.5$, $P_f = 0.1$, $P_d = 0.9$ and $N = 20$. Both results were gathered from 300 simulations. Figure 2.3 shows the comparison between the MAP estimator and the system with dedicated channels between the local sensor to DFC (sensor-DFC) (calculation based on [26]). We observe with full-state information, the MAP estimator appears to converge to that of a distributed detection system with dedicated sensor-DFC channels as the size of contention window increases since all the local sensors are able to communicate with the DFC when the window size is long enough. When the channel state was not fully available (Figure 2.4), the performance of the EM estimator did not converge to the optimum even when the contention window size reached 2000. Still, there was marked improvement over the performance of a single sensor.

In Figure 2.5 we compare the probability of error gathered from our MAP estimator to the probability of error gathered from Yuan's CRA algorithm [68] for a system with $N = 10$, $P_f = P_m = 0.1$ and $P_0 = P_1 = 0.5$. Probabilities of error are shown for the system with Yuan's CRA (simulation), with the CRA proposed in this paper, and with dedicated sensor-DFC channels (calculated based on [26]). We observe that the advantages of our algorithms in terms of implementation reduced our performance, we take much longer time to converge to the optimal solution.

In Figure 2.6 we show the transmission probability of a specified sensor against time for a distributed detection system with the configurations shown in Table 2.1. The dashed

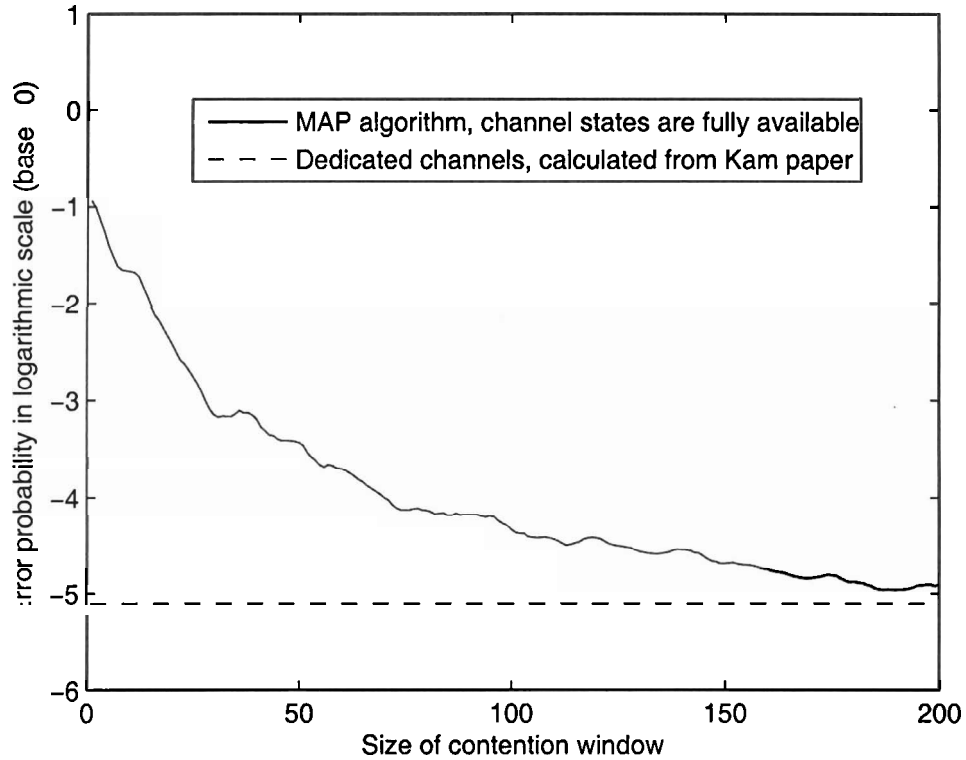


Figure 2.3: Comparison of error probability (MAP)

line represents the analytical result of transmission probability calculated by Equation 2.33. The solid line represents a simulation of the transmission probability. We observe that after 2000 time slots, the simulation result appears to have converged to the analytical result. Figures 2.7, 2.8, and 2.9 show the corresponding probabilities of *success*, *idle*, and *collision* states of the communication channel, respectively.

In order to study the accuracy of these estimators in estimating the number of transmitting sensors, we use a root-mean-square-error (RMSE) metric. The RMSE for each simulation time is calculated as follows:

$$RMSE = \frac{1}{N_{sim}} \sum_{i=1}^{N_{sim}} \sqrt{\frac{1}{N} \sum_{n=1}^N (\tilde{n} - n)^2}, \quad (2.70)$$

where N_{sim} is the total simulation time, N is the total number of sensors connected to the

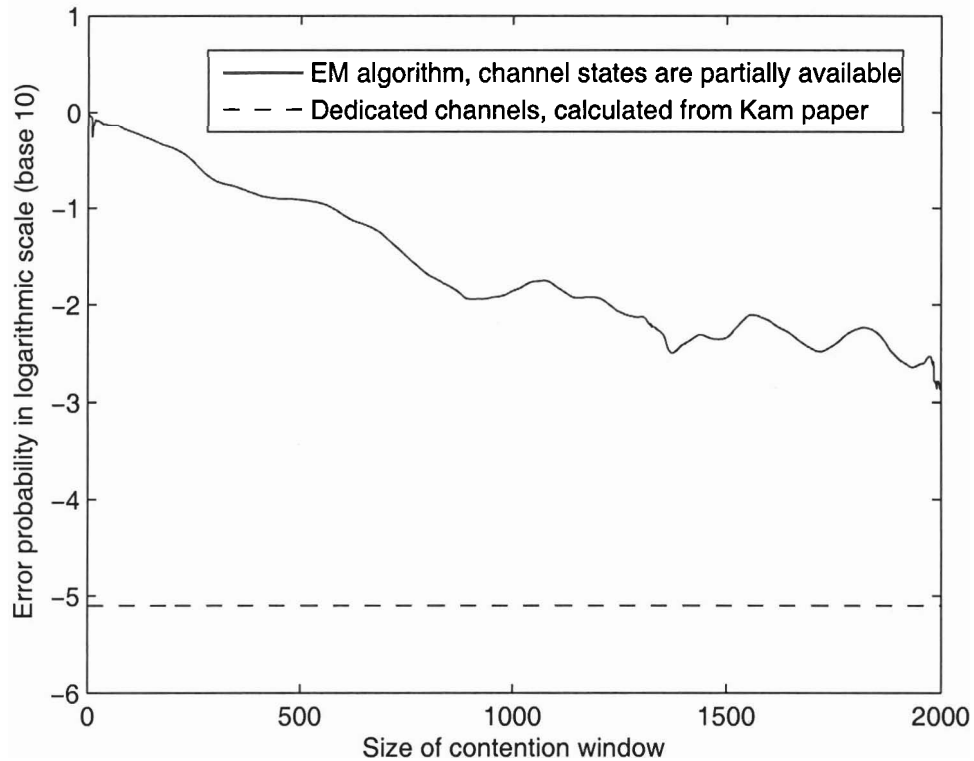


Figure 2.4: Comparison of error probability (EM)

channel, and \tilde{n} is the estimate of the number of transmitting sensors n .

Figure 2.10 shows the RMSE of the MAP estimator for the number of transmitting sensors against the size of contention window under full state information. Figure 2.11 shows the RMSE of the EM estimator when only partial state information is available. In both of the cases, we set $P_0 = P_1 = 0.5$, $P_f = 0.1$, $P_d = 0.9$, $N = 20$ and the simulation is repeated 300 times. As expected, the RMSE of the estimator decreases as the size of contention window increases. The steady state error of the MAP estimator with full state information is smaller than the steady state error of the EM estimator (with partial state)

Figure 2.12 gives the receiver operating characteristic (ROC) curves obtained by using the K-out-of-N optimal fusion rule with dedicated sensor-DFC channels. In this example, the total number of sensors is 20 with sensor level $P_f = 0.3$ and $P_d = 0.7$. The tradeoff

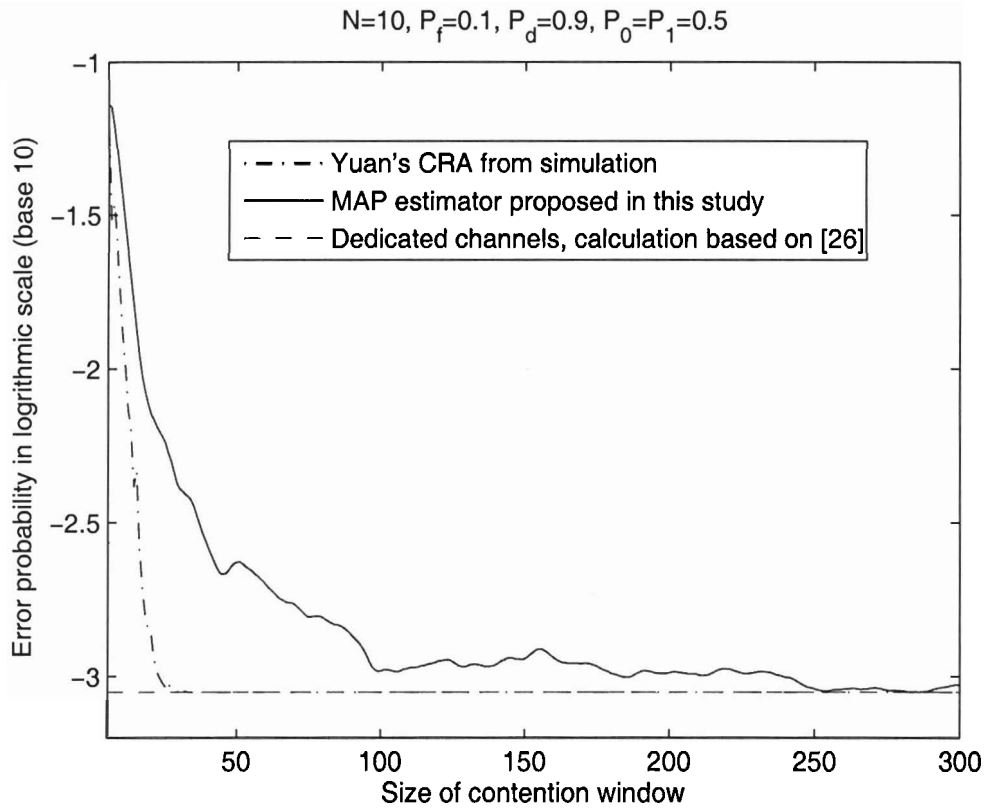


Figure 2.5: Comparison of error probability with the CRA proposed in [68]

between the probabilities of detection and false alarm for the different *a priori* probabilities is also shown in the figure.

2.6 Conclusion and Discussion

A distributed detection system using a single random access communication channel is studied in this chapter. Based on our sensor model that incorporates the collision avoidance mechanism to transmit local decisions to the DFC, two decision schemes are developed for estimating sensor transmissions on the channel under different channel conditions. These schemes are geared toward sensor field applications where pre-warnings are sent over the shared channel to inform of the presence of threats, and where resources are limited and central processing is impossible. Simulations show that the system performance matched

Parameters	Values
N	20
n	10
W_{base}	10
m	20
W	150
Duration of Simulation	10000
Simulation Times	300

Table 2.1: The set of parameters used in simulation

what we predicted by using the two dimensional Markov-chain sensor model. For the decision schemes, the first scheme is developed using MAP estimator under the assumption that the channel statistics are fully available to the DFC. While the second scheme is developed using EM estimator by assuming only partial channel statistics are available to the DFC. Simulations also show that the performance of MAP estimator is always better than EM estimator from the aspects of RMSE and the needed size of contention window due to the different degrees of channel statistics awareness.

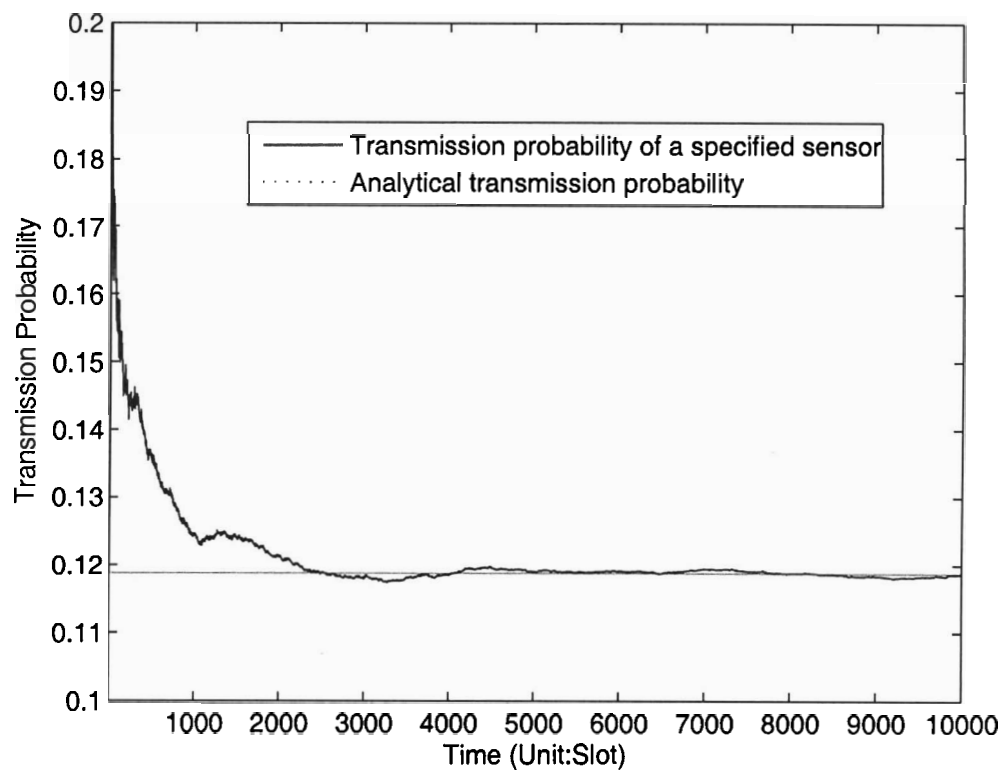


Figure 2.6: Probability of transmission of a specified sensor vs. Time under the conditions in Table 2.1

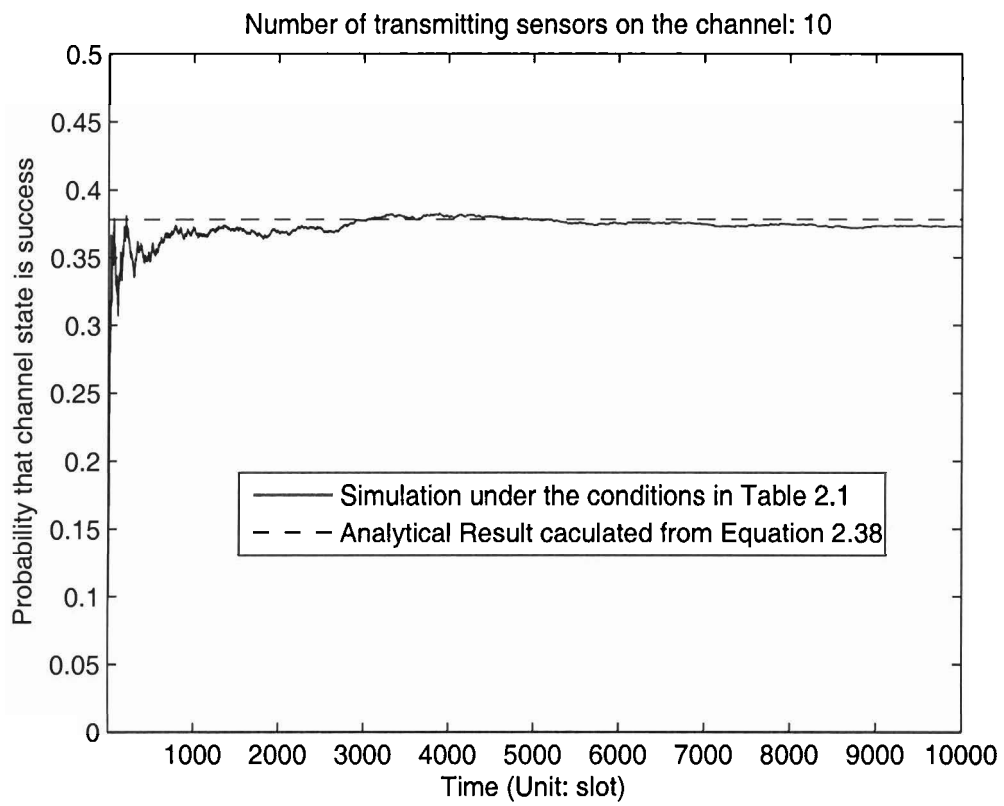


Figure 2.7: Probability of Successful transmission vs. Time under the conditions in Table 2.1

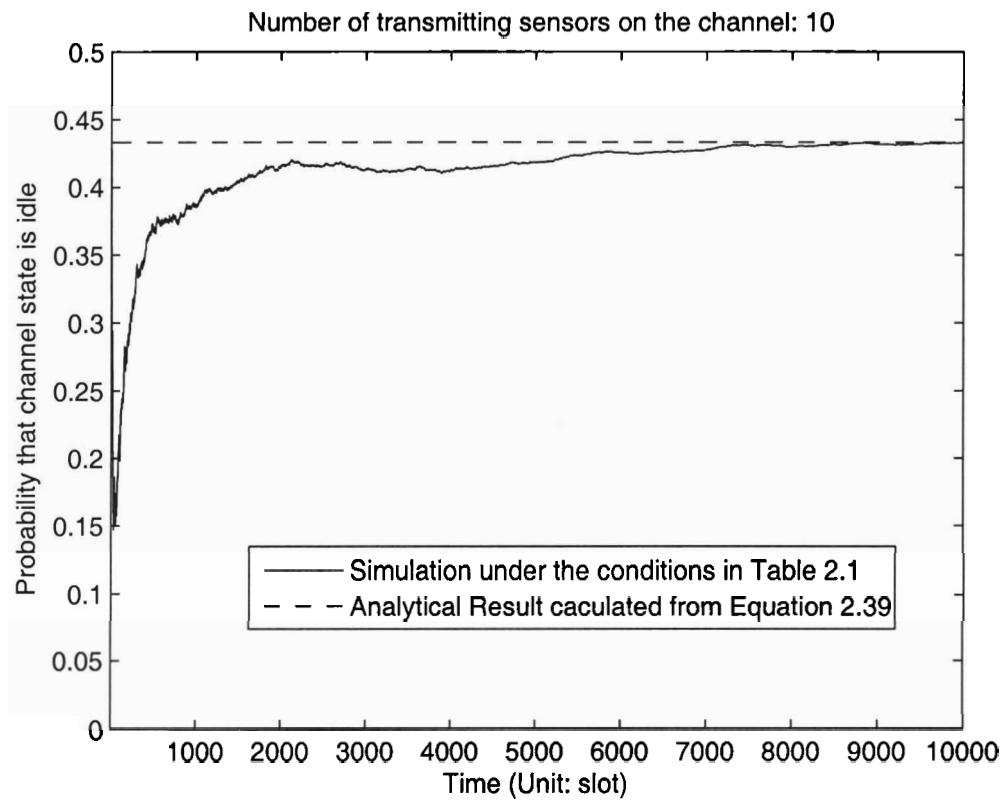


Figure 2.8: Probability of Idle state vs. Time under the conditions in Table 2.1

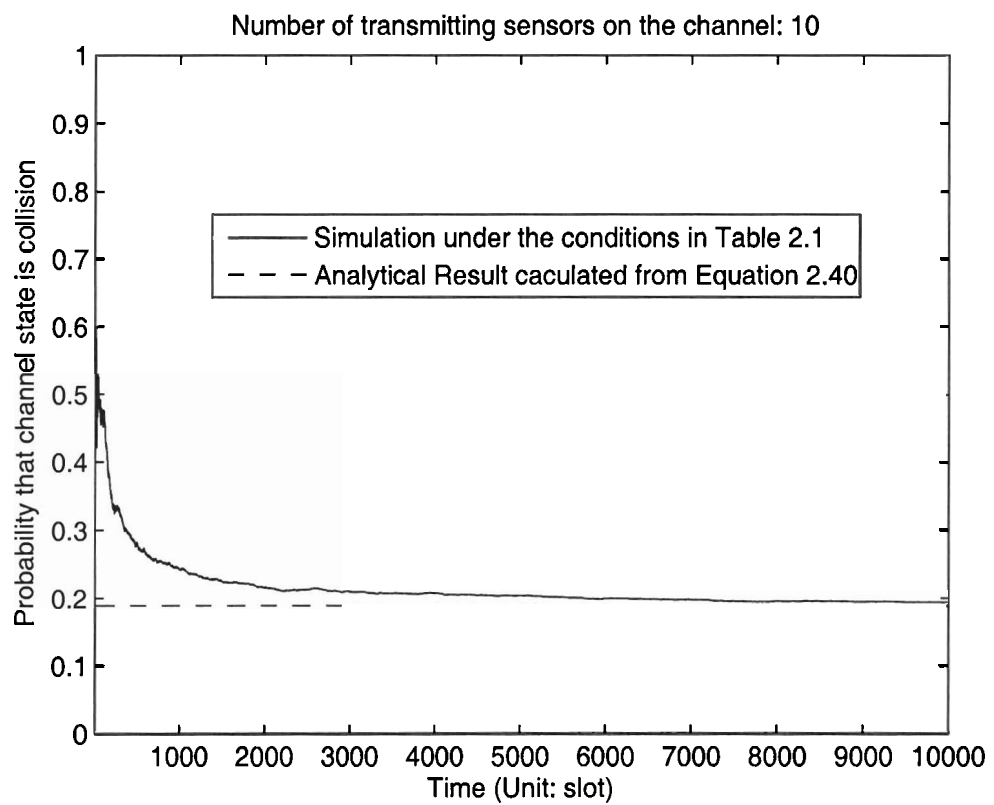


Figure 2.9: Probability of collision state vs. Time under the conditions in Table 2.1

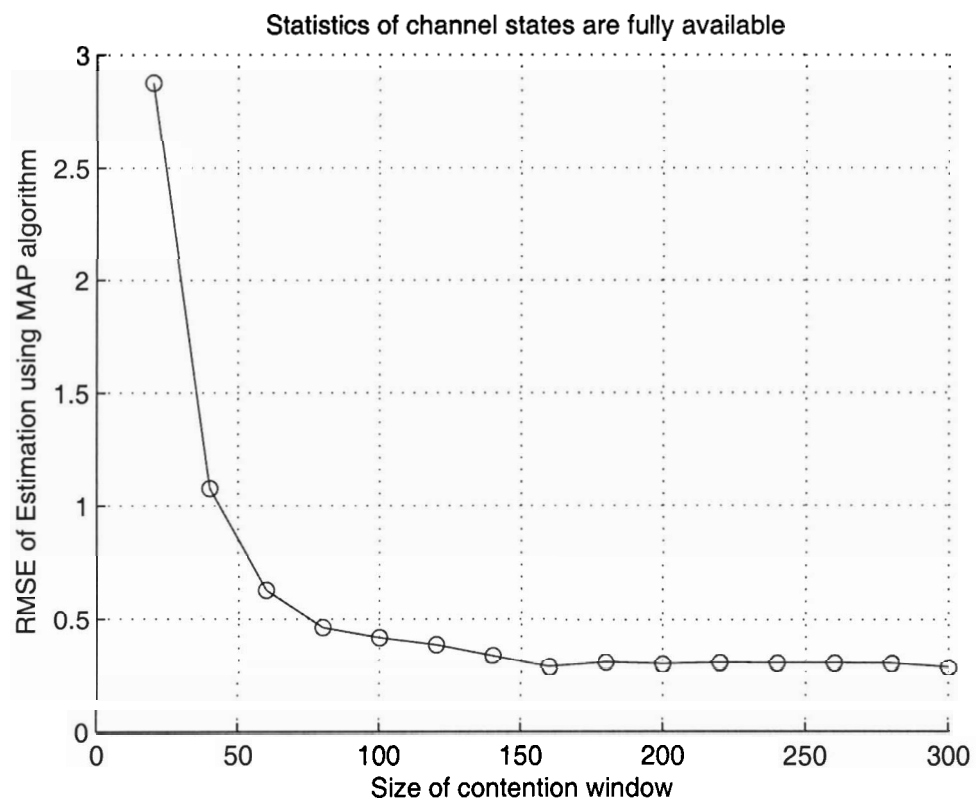


Figure 2.10: RMSE of the MAP estimator when channel states are fully available

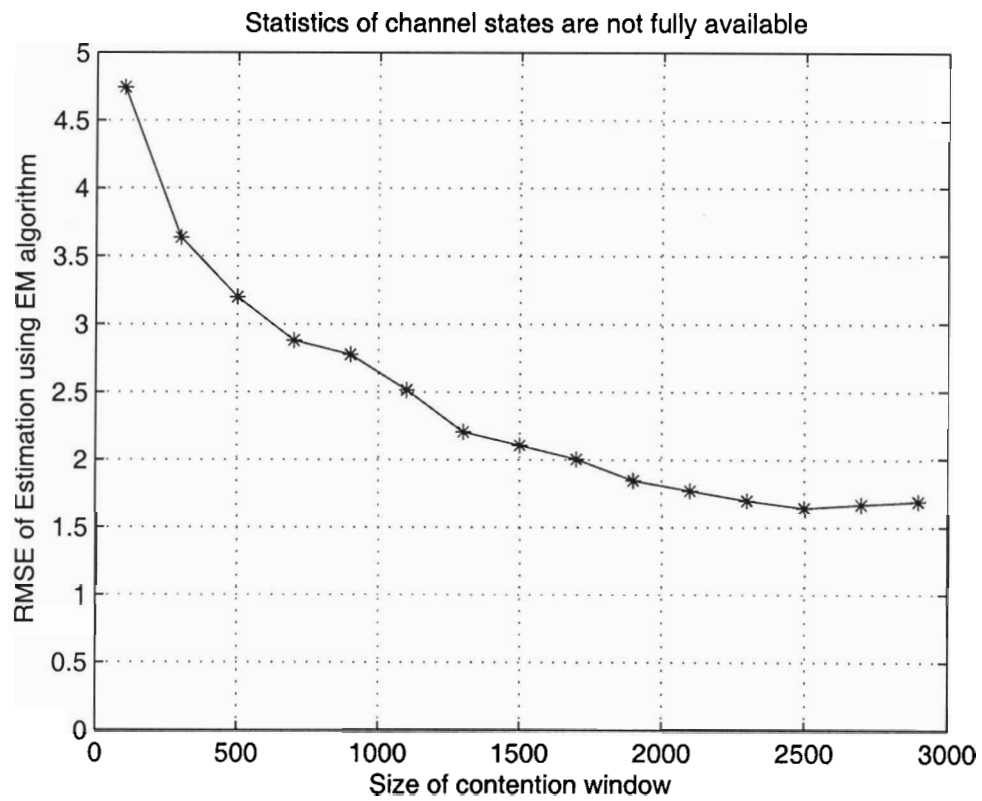


Figure 2.11: RMSE of the EM estimator when channel states are not fully available

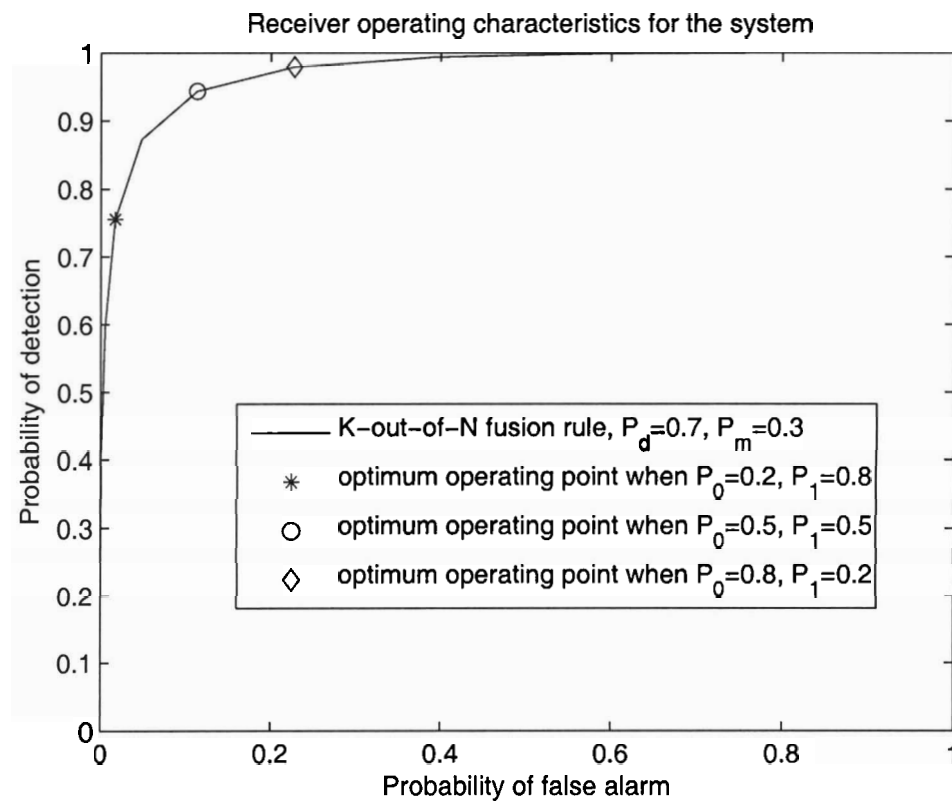


Figure 2.12: Receiver operating characteristics for the system. $N = 20$, $P_d = 0.7$, $P_f = 0.3$

Chapter 3. Distributed Detection in LonWorks Application

3.1 Introduction

In this chapter we use the approach developed in Chapter 2 to analyze a commercial application of distributed detection for control applications. Specifically, we study the Carrier Sense Media Access (CSMA) protocols, which are used to control the access of a computer network device to a shared channel in distributed control architectures. CSMA protocols belong to the family of contention protocols, which allow sensors to compete for network access. CSMA protocols are used in several networking standards (*e.g.*, IEEE 802.3 CSMA/CD (ethernet) standard [1], IEEE 802.11 wireless LAN standard [5]), and come in several variants (non-persistent, 1-persistent, p -persistent). In this chapter, we concentrate on the predictive p -persistent CSMA protocol, which was proposed by the Echelon Corporation in the 1980s (registered as the American standard ANSI/EIA-709.1 [6] and European standard ENV 13154-2). The predictive p -persistent CSMA protocol was proposed for use in distributed sensor networks and is used in the Media Access Control sublayer of the LonTalk protocol in Local Operating Networks (LonWorks) [2].

In most contention algorithms, terminals reschedule transmission of packets that were unsuccessfully transmitted before according to a randomly distributed retransmission delay. In p -persistent CSMA [28, 54], the time window during which the next packet will be transmitted is finely slotted. If a terminal is ready to transmit a packet, and if it senses that the channel is idle, it would transmit the packet during the next slot with probability p . It would delay transmitting the packet with probability $1 - p$ by the equivalent of one slot. If the terminal detects that the channel is idle at this point in time, it would repeat the process. Otherwise, another terminal has begun transmission and our terminal will reschedule retransmission according to a retransmission delay distribution. If the ready

terminal senses the channel busy, it would wait until it senses that the channel is idle and then operate as above.

The difference between the p -persistent CSMA algorithm as introduced in 1975 and the predictive p -persistent CSMA algorithm is that in the former the probability p was constant, while in the latter p depends on the channel backlog. Past evaluations of the predictive p -persistent CSMA algorithms (Chen [16], Miskowicz [41]) relied primarily on simulations (not analytical models), and provided no experimental verification of their predictions. Here we supplement these past evaluations by developing an analytical model for the predictive p -persistent CSMA algorithm and by providing experimental results from a six-sensor network of *SMART I/O ADR112-F* units [4] connected by TP/FT-10 media-type network [2]. Our analytical model is inspired by performance analysis of the 802.11 distributed coordination function by Bianchi [11].

The rest of the chapter is organized as follows: In section 3.2 we review the predictive p -persistent CSMA algorithm as implemented by the LonWorks protocol. In section 3.3 we present a probabilistic calculation of the p -persistent CSMA algorithm, following by a Markov-chain model of the predictive p -persistent CSMA algorithm. The probabilistic calculation is from the “channel viewpoint”, while the Markov-chain model is “sensor-centric”. In section 3.4, we describe the implementation details of the simulation tool by using the *OPNET Modeler* [3]. Section 3.5 validates the performance of the simulation tool by comparing results gathered from simulations, from a physical system using LonWorks sensors, and from the analytical models.

3.2 An Overview of the Basic Mechanism of the LonWorks Protocol

3.2.1 The predictive p -persistent CSMA algorithm implementation in LonTalk

The LonTalk protocol([EIA-709.1-A]) is a peer-to-peer networking protocol that employs a collision resolution algorithm (CRA) to allow sensors to share a single time-slotted channel

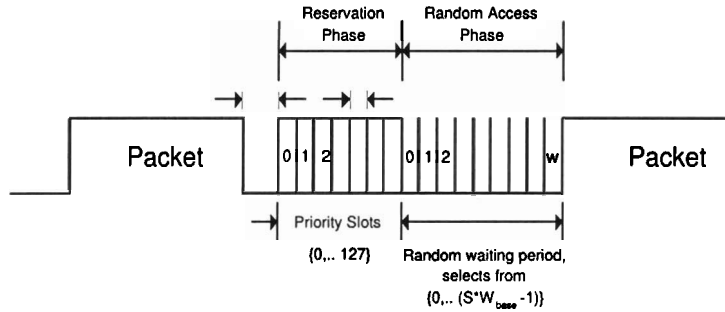


Figure 3.1: Packet cycle of LonWorks protocol

by using knowledge of the expected channel load. Like some other existing CSMA algorithms, the LonTalk MAC algorithm (the predictive p -persistent CSMA algorithm) splits the time axis into segments called slots, whose length is specified as β_2 . Figure 3.1 shows the packet cycle of the LonTalk protocol. A *reservation phase* is used for priority messages, and a *random access phase* is used for all other messages. The size of the randomizing window used in the second phase depends on the backlog, and can extend from 16 to 1008 time slots (each of a specified length β_2). Here we assume that the priority slots are not used, and any new packets will be transmitted between 0 and $w\beta_2$. Specifically we show a packet transmitted in slot $w + 1$, where w is the random waiting period as described below: At any given time, the communication channel is in one of the following three states: *idle*, *collision*, *communication*. Each sensor connected to the channel, when it has a packet ready for transmission, follows the algorithm as shown in Figure 3.2. The sensor monitors the state of channel, and classifies the channel as *idle* if it detects no transmission during a period of length β_1 . Next, the sensor starts a *transmission cycle*, which depends on two positive integers, w and W , and $w < W$. The next transmission would occur within a window from 0 to $W\beta_2$, and the sensor would attempt transmission only after a waiting period of $w\beta_2$ and only if it senses that the channel is idle after the waiting period. The value of w is selected uniformly at random from the set $\{0, 1, \dots, W\}$, and the value of W is chosen as

$$W = (S \times W_{base}) - 1, \quad (3.1)$$

where S , and W_{base} are positive integers. In the LonTalk protocol, S represents as current backlog size (initially $S = 1$) and $S \in \{1, 2, \dots, 63\}$, $W_{base} = 16$. If after waiting for $w\beta_2$ the channel is not idle, or if it is idle but a collision follows the transmission attempt, the sensor will start a new transmission cycle. In the latter case (collision) it would increment S by 1 as long as $S < 63$.

3.2.2 Message services

To ensure that messages are being transmitted successfully via the communication channel, LonWorks offers two message services for packet transmission: 1) *Acknowledged message service* (ACKD) provides for end-to-end acknowledgement. When using this service, a message is sent to a device or group of up to 64 devices and individual acknowledgements are expected from each receiver. If acknowledgements are not received after a specified timeout period, the sender start a new transmission cycle. 2) *Unacknowledged message service* (UNACKD_RPT) causes a message to be sent to a device or group of any number of devices multiple times and no acknowledgements are expected. This message service has the lowest overhead and is the most typically used service. Since none of the packets are acknowledged the value of S remain fixed at 1, and the protocol is non predictive (the original p -persistent CSMA algorithm [28]).

3.3 Modeling of the LonWorks Communication Protocol

To study the performance of the LonWorks network, we modeled the LonWorks channel behavior in two modes of operations: with unacknowledged service, and with acknowledged service. The model for unacknowledged service is developed from the viewpoint of the communication channel, and is essentially for non-predictive p -persistent CSMA.

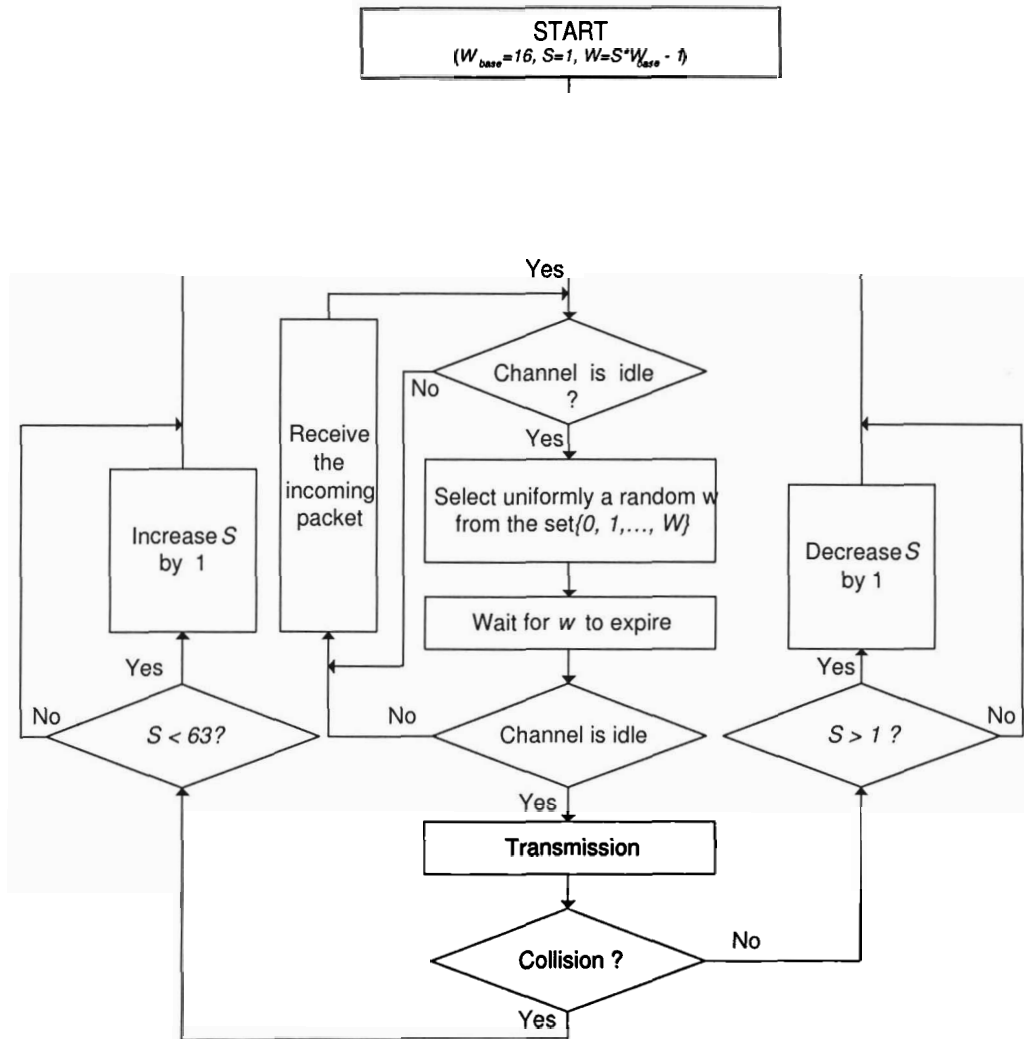


Figure 3.2: Block diagram of the p -CSMA algorithm

Related work can be found in [16] and [41]. We derive the probability of the collision rate for the channel. The model for acknowledged service is developed from the viewpoint of the sensor under the assumption that the randomizing window of each sensor is dynamically changed according to the probability of message collision. In developing this model, for the predictive p -persistent CSMA algorithm, we relied on the expanded CRA model developed in Chapter 2.

3.3.1 Model for p -persistent CSMA (Unacknowledged service in LonTalk protocol)

In this case $S = 1$ and $W = 15$. We use N to designate the number of sensors connected to the channel. When the sensor with the smallest value of w gains access to the channel (provided it is the only sensor that selected this value of w) it will try to transmit. Assuming the network is operating under a heavy steady state load and N local sensors are always competing for the channel, a single sensor would gain access to the channel during the w th time slot with probability

$$P_s^{(w)} = N \frac{1}{W+1} \left(\frac{W+1-w}{W+1} \right)^{N-1}. \quad (3.2)$$

The probability that no sensor would attempt to get access to the channel during the w th time slot is

$$P_i^{(w)} = \left(\frac{W-w+1}{W+1} \right)^N. \quad (3.3)$$

The probability of a collision during w th time slot is

$$P_c^{(w)} = 1 - P_s^{(w)} - P_i^{(w)}. \quad (3.4)$$

The probability that a collision occurred during the W window after the channel became available is:

$$P_{ch.coll} = \sum_{w=0}^W P_c^{(w)} \quad (3.5)$$

3.3.2 Model for predictive p -persistent CSMA (Acknowledged service in LonTalk protocol)

We develop the model for predictive p -persistent CSMA assuming the worse case, namely “saturation throughput”. In saturation, every sensor that is connected to the bus channel always has a packet available for transmission immediately after the completion of each

successful transmission. We further assume that we are in steady state: whenever a sensor is about to transmit a packet (having waited w time slots) its probability of successful transmission (neither busy channel nor collision) is q .

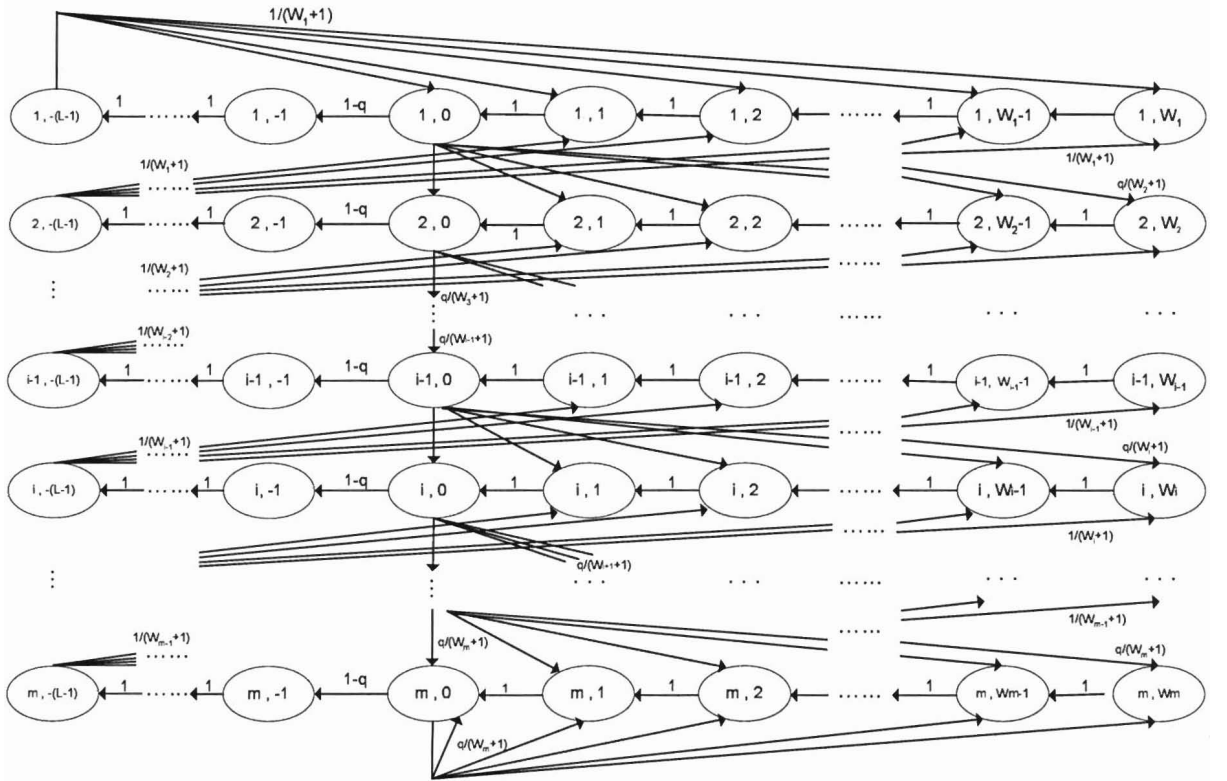


Figure 3.3: Markov chain model for the backlog window size of LonWorks sensor

For any give sensor, let $b(t)$ be the stochastic process that represents the time left until the end of the current random waiting period, $b(t) \in \{0, 1, \dots, S_{max} \times W_{base} - 1 (= 63 \times 16 - 1 = 1007)\}$. A discrete, integer time scale is adopted where t and $t + 1$ correspond to two consecutive slot times, and the backlog time counter of each station decreases at the beginning of each slot time. We also designate $s(t)$ be the stochastic process that represents the backlog coefficient (S in Equation 3.1, $s(t) \in \{1, 2, \dots, S_{max} (= 63)\}$). As Figure 3.2 shows S is incremented or decremented after each transmission attempt.

The key assumption of this model is that a packet transmission fails with the same probability q regardless of the backlog time counter used for this transmission. Based on this assumption, we modified Bianchi's discrete time Markov chain model [11], which was originally developed to study the MAC algorithm of IEEE 802.11 protocol for wireless local area networks. We can model $s(t), b(t)$ by the discrete time Markov chain model presented in Figure 3.3. We show in the figure that the Markov chain has m rows. The s th row has $W_s + L$ states, here $W_s = sW_{base} - 1$ and L is the average length of packets to be transmitted on the channel. Each row corresponds to one of the 63 states of $s(t)$, and each row has $16s + L - 1$ states, representing all possible waiting times to transmission attempts ($b(t)$).

At the beginning of a transmission cycle, a sensor whose backlog is $s(t)$ is in state (s, w) , and its state moves to the left every time step (to $(s, w - 1)$, $(s, w - 2)$, etc.) until it reaches $(s, 0)$. When the state reaches $(s, 0)$, the sensor either (a) decides not to transmit since the channel is busy, or transmit and face a collision; or (b) transmits successfully. In case (a) it moves at the next time slot to a state in row $s + 1$ (except if s was 63) and uniformly chooses a new w from a larger range size of W (W_{base} slots more). In case (b) it moves to the left until it reaches $(s, -(L - 1))$, which means the transmission is completed and the sensor moves at the next time slot to row $s - 1$ (except if s was 1).

If we adopt the short-hand notation $P\{i, j|k, l\} = P\{s(t + 1) = i, b(t + 1) = j | s(t) = k, b(t) = l\}$ for the conditional transition probabilities, the one-step transition probabilities

can be found as

$$P\{i, j|i, j + 1\} = 1, \quad (3.6)$$

$$i \in (1, m), \quad j \in (0, W_i - 1) \text{ or } j \in (-2, -(L - 1));$$

$$P\{i, -1|i, 0\} = 1 - q, \quad (3.7)$$

$$i \in (1, m);$$

$$P\{i, j|i + 1, -(L - 1)\} = \frac{1}{W_i + 1}, \quad (3.8)$$

$$i \in (1, m - 1), \quad j \in (0, W_i);$$

$$P\{i + 1, j|i, 0\} = \frac{q}{W_{i+1} + 1}, \quad (3.9)$$

$$i \in (1, m - 1), \quad j \in (0, W_{i+1});$$

$$P\{1, j|1, -(L - 1)\} = \frac{1}{W_1 + 1}, \quad (3.10)$$

$$j \in (0, W_1);$$

$$P\{m, j|m, 0\} = \frac{q}{W_m + 1}, \quad (3.11)$$

$$j \in (0, W_m).$$

Equation 3.6 states that after the sensor generates the random waiting timer according to the estimated backlog window size, the timer keeps decreasing until it reaches $(i, 0)$. If the transmission of the packet at time $(i, 0)$ is successful (probability is $1 - q$), the sensor moves one state toward $(i, -(L - 1))$ and keeps transmitting the packet until it reaches the state $(i, -(L - 1))$, then the sensor moves one stage toward stage 1 and the estimated backlog window size is decreased by 1. The w value for the transmission of the next packet will be generated uniformly from a smaller range $(0, W_i)$, which is represented in Equation 3.8. Equation 3.9 shows that if the transmission of the packet at state $(i, 0)$ has failed, which means that the transmission is unsuccessful (a collision has occurred or the channel is busy) with probability q , the sensor moves one step toward stage m and the estimated backlog window size is increased by 1. The new selected w will be generated from a larger range $(0, W_{i+1})$. Equation 3.10 considers the special condition of the sensor being at stage 1

when it transmits a packet successfully. Equation 3.11 is the special case when the sensor is at the last stage and it fails to transmit a packet.

We designate $b_{i,j} = \lim_{t \rightarrow \infty} P\{s(t) = i, b(t) = j\}$, $i \in (1, m)$, $j \in (0, W_i)$ to be the stationary behavior after the process has been running for a long time. After some algebra and applying the method of induction, it can be shown that

$$b_{i,j} = \frac{W_i + 1 - j}{W_i + 1} b_{i,0}, \quad i \in (1, m), \quad j \in (0, W_i), \quad (3.12)$$

and

$$b_{i,j} = (1 - q)b_{i,0}, \quad i \in (1, m), \quad j \in (-1, -(L - 1)), \quad (3.13)$$

where

$$b_{i,0} = \left(\frac{q}{1 - q}\right)^{i-1} b_{1,0} \quad (3.14)$$

Since the sum of the probabilities of all the states in a Markov Chain model is unity, we have,

$$\sum_{i=1}^m \sum_{j=-(L-1)}^{W_i} b_{i,j} = 1 \quad (3.15)$$

We can further derive by substituting $b_{i,j}$ from Equation 3.12, 3.13 and $W_i = iW_{base} - 1$ into Equation 3.15

$$b_{1,0} = \frac{1}{(L - 1)(1 - q) \sum_{i=1}^m \left(\frac{q}{1 - q}\right)^{i-1} + \frac{W_{base}}{2} \sum_{i=1}^m i \left(\frac{q}{1 - q}\right)^{i-1}} \quad (3.16)$$

We can now express the probability τ that a sensor transmits in any randomly chosen time slot. As any transmission occurs when the backlog time counter is equal to zero, regardless of the backlog stage,

$$\tau = \sum_{i=1}^m \sum_{j=-(L-1)}^0 b_{i,j}. \quad (3.17)$$

After substituting and applying the equation of Arithmetic-Geometric series, τ can be expressed as

$$\tau = \frac{L(1 - q) - q}{(L - 1)(1 - q) + \frac{W_{base}[(2m+1)q^{m+1} + (1-q)^{m+1} - (m+1)q^m]}{2[(1-q)^m - q^m](1-2q)}} \quad (3.18)$$

The probability τ depends on the probability q , which is still unknown. The probability q that a transmission attempt was aborted or resulted in collision is the probability that at least one of the remaining $N - 1$ stations transmit in the same time slot. Hence, we are able to get another equation for p and τ :

$$q = 1 - (1 - \tau)^{N-1} \quad (3.19)$$

Equations 3.18 and 3.19 form a nonlinear equation set with unknowns q and τ . The system can be solved by numerical methods. It is easy to prove that there is a unique solution to this nonlinear system since, in Equation 3.18, τ is a continuous and monotonically-increasing function in the range of $(0, 1)$. In Equation 3.19, τ is a continuous and monotonically-decreasing function in the same range. Which means that, a point of intersection between 0 and 1 can always be found.

Let P_{tr} be the probability that at least one transmission attempts occurs in the given time slot. For a LonWorks network with N sensors, P_{tr} is given by

$$P_{tr} = 1 - (1 - \tau)^N \quad (3.20)$$

The probability P_{succ} , that an occurring transmission in a given time slot is successful, is given by the probability that one sensor transmits and the remaining $N - 1$ sensors remain idle given that at least one transmission occurs on the channel:

$$P_{succ} = \frac{N\tau(1 - \tau)^{N-1}}{P_{tr}} \quad (3.21)$$

Hence, a successful transmission in a given time slot of the communication channel will be $P_{succ}P_{tr}$, and with probability $(1 - P_{succ})P_{tr}$ the given time slot of the communication channel contains unsuccessful attempts.

3.4 Integrated Simulation Model

3.4.1 Design concepts

In this section, we describe the methodology and tools [33, 64] used to conduct the performance evaluation of the LonWorks protocol. The simulation environment consists of detailed modules for the communication channel, MAC layer, Network and Transaction layer, Transport layer, and Application layer developed in *OPNET Modeler* [3]. These detailed simulation models will constitute the evaluation framework for studying the data traffic conditions on the LonWorks communication channels. Some parts of the LonWorks protocol (*e.g.*, Session layer and Link layer) have been simplified, omitted or deferred since it is intended primarily to study and evaluate the performance of the LonWorks protocol at the data-networking level.

Figure 3.4 shows the OPNET network model we have developed for the LonWorks protocol. The figure illustrates a simple bus network channel (TP/FT-10) with five sensors, along with the corresponding hierarchical relationship with an inside view of the sensor and process modules. The left-hand top corner shows a network of five sensors connected to a single multiple access channel. Each unit is then expanded to reveal its internal structure.

3.4.2 Supported functionalities

Table 3.1 summarizes the functions supported in the simulation model. Our model assumes that no propagation delays occur on the communication channel, and only transmission delays are considered. Moreover, all stations are deferred for an integral number of β_2 time slots. Therefore, collisions can occur only at the beginning of the transmission cycle. We

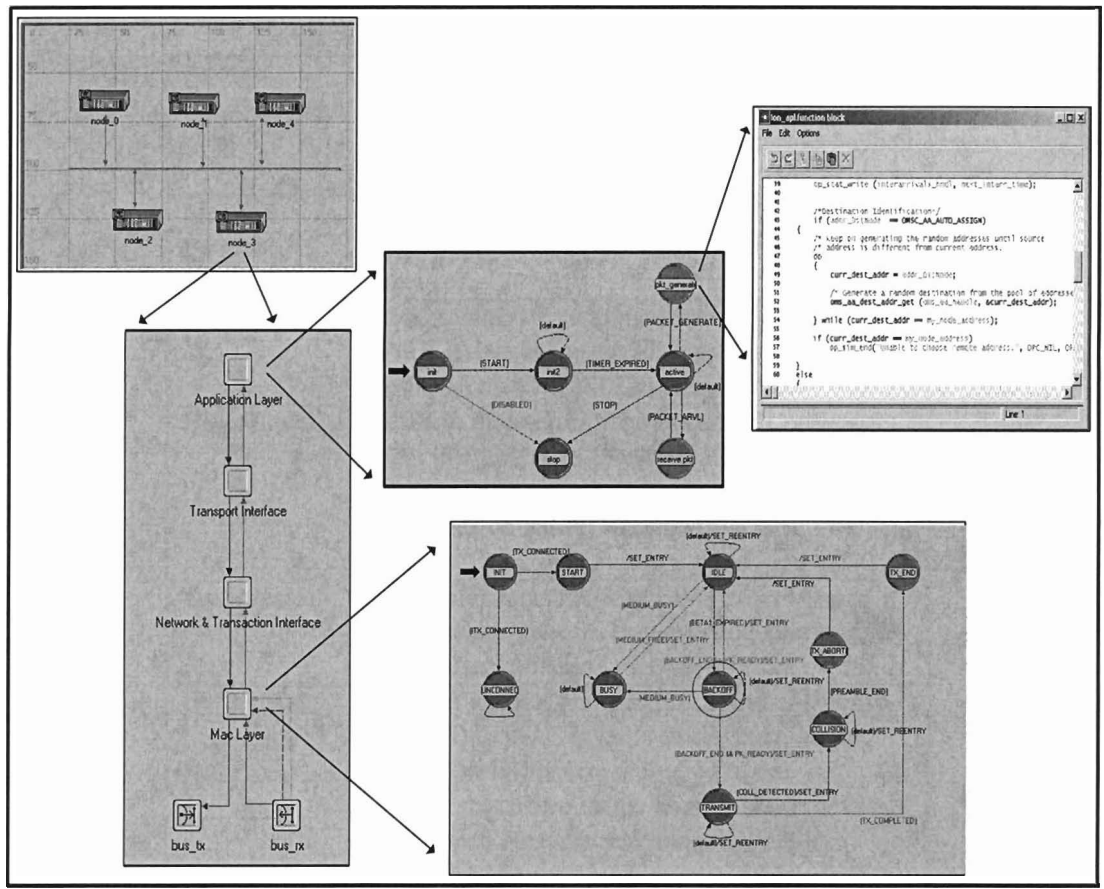


Figure 3.4: Hierarchical structure of network model

assume that packet error is caused only by collisions, and that packet loss is a result of the buffer between the MAC layer and higher layers being full. In other words, the rest of the channel is ideal.

3.4.3 Node model

This section gives an operational description of the LonWorks node model implemented in OPNET. The node model consists of the following modules:

Table 3.1: Summary of supported functionalities

Access Mechanism	Carrier senses multiple access with collision avoidance (CSMA/CA) and optional collision detection (CSMA/CD) access schemes as defined in the standard.
Message Service	Two kinds of message services are supported: acknowledged messages and unacknowledged messages.
Data Rate	78Kbps data rate supported by LonWorks TP/FT-10 bus channel 1.25Mbps data rate supported by TP/XF-1250 bus channel
Input Clock	Different input clock frequencies of LonWorks node: 0.625MHz, 1.25MHz, 2.5MHz, 5MHz, 10MHz. Different clocks have different limitations on packet rate and time slot duration.
Transmission	Full-duplex transmissions. FIFO processing of transmission requests. Transmission attempt limit of 256 after collision.
Buffer Size	A packet that has arrived from a higher layer to the LonTalk MAC layer is stored in an output buffer. The buffer size is limited to the adjusted maximum value. Higher-layer packets are dropped once the maximum buffer size is reached.
Node Auto-addressing	All the nodes can be configured for node ID auto-addressing. User can also specify node addresses. However, no subnet or group address is supported yet. Destination address of each packet can be chosen automatically from address pool of all the nodes.

3.4.3.1 MAC module

As we mentioned in Section 3.1, LonWorks employs p -CSMA as its media access protocol in the MAC layer. We have designed a finite state machine (FSM) for the MAC layer according to the description of predictive p -CSMA algorithm (Section 3.2) and integrated it into the MAC module.

In the FSM of the MAC module, operation starts from the INIT state, which is used to initialize the source parameters for the simulation and ensure the node is connected to

the communication channel. If it is connected to the network, the MAC module goes to the IDLE state. In the IDLE state, the module monitors the state of the channel, and determines if the channel is still in the IDLE state. If it detects that the channel is busy, it will enter the BUSY state. Otherwise, if it detects no transmission during the β_1 period, it will enter the BACKLOG state, where a w is generated in the interval $\{0, 1, \dots, W_{base} \times S - 1\}$ and the module waits for it to expire. When the waiting period expires, nodes without a packet to transmit will go back to IDLE and remain in synchronization, and nodes with a packet to transmit enter the TX state if the communication channel is still idle when rwp expires. If the transmission is successful, the node enters the TX_END state. Otherwise, it goes to the COLLISION and TX_ABORT states. The whole process repeats until there are no more packets arriving at the MAC layer and requiring transmission .

Each node maintains an estimate of the backlog window size W throughout the operation by monitoring the following conditions: I) The MPDUs it sends or receives, II) If a collision is detected, III) If a packet cycle or W_{base} randomizing slots go by without channel activity.

3.4.3.2 Network and Transaction module

There are two services provided in this module, namely I) address recognition, II) duplicate transaction detection. Each LonWorks node performs address recognition during the operation. The module receives and forwards the message from MAC layer to the upper layer if the address (or Unique_Node_ID) of the message matches the address of the current node. To make sure the receiver only acts upon the same packet once, the transaction control algorithm is established to perform duplicate detection.

The transaction control algorithm uses 4-bit transaction numbers that are initialized by the sender to guarantee the ordering among outgoing messages and are used by the receiver to detect duplicate packets. The detailed implementation of the transaction control

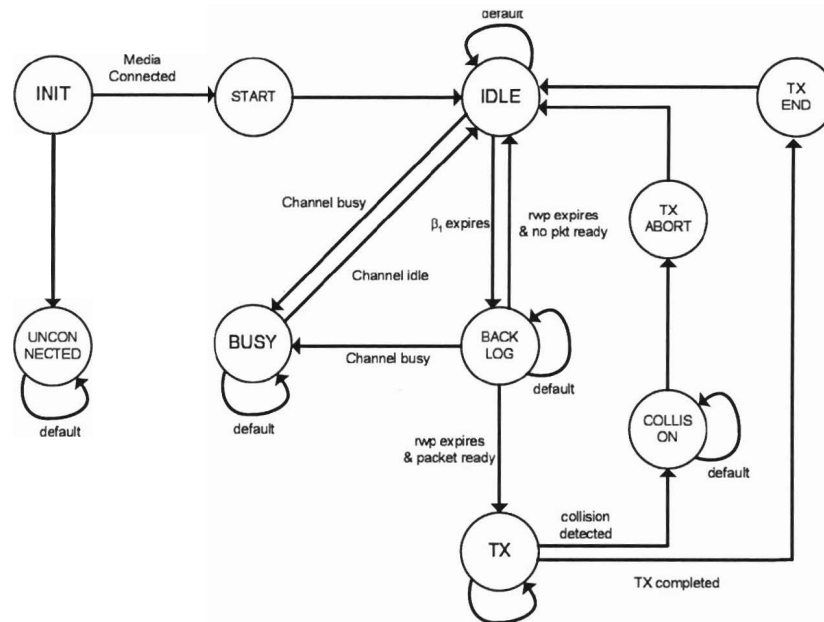


Figure 3.5: Finite state machine of MAC module

algorithm follows the following basic operations

- When a node has a packet to send, it picks a transaction ID from $(0, 1, \dots, 15)$
- Upon receipt of that packet, the receiver decides whether the packet is a duplicate of a previous packet or a new one according to the attributes stored in TPDU.
- If the packet is a new one, the receiver allocates a new record and starts a receive transaction timer for the packet (In our simulation, the receive transaction timer is 8 seconds)
- If the packet is detected as a duplicate, it will not be delivered to the upper layer. The process model for the network and transaction module is shown in Figure 3.6.

3.4.3.3 Transport module

Figure 3.7 shows the process model of the transport module. In the INIT and WAIT states, the module initializes its state variables used in the entire process. Then the node registers

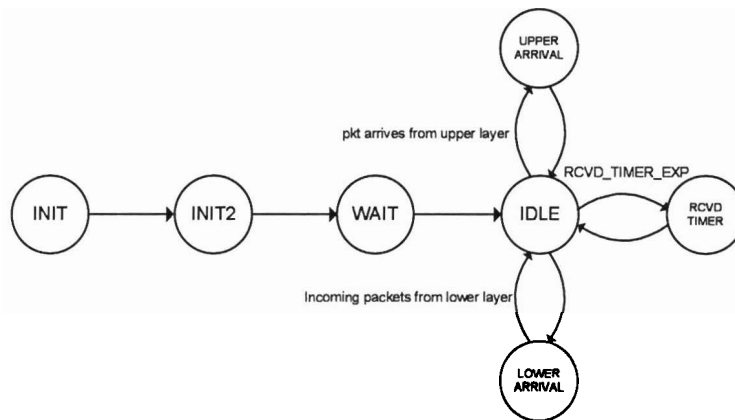


Figure 3.6: Finite state machine of Network and Transaction module

itself and makes connections with other process nodes that are connected to the transport module. After leaving the initial states, the module switches among IDLE, XMIT_EXPIRES, and two packet processing states (UPPER_ARRIVAL and LOWER_ARRIVAL). The module enters the IDLE state and waits for an incoming event. The event can be either an incoming packet from the application module, an incoming packet from the network and transaction layer, or the expiration of the retransmission timer Xmit_Timer when using acknowledged message service. When a packet arrives from the application module, the UPPER_LAYER_PKT_ARVL event is triggered, and the state machine enters the UPPER_ARRIVAL state where the type of packet is determined and the required processing and encapsulation are executed. Then the packet is forwarded to the network and transaction layer where it will be enclosed into an NPDU.

During the operation, if an unacknowledged message service is selected, the module only handles packets from two directions and switches the states among IDLE, UPPER_ARRIVAL and LOWER_ARRIVAL. If acknowledged messages are being used, every time the module receives a packet from the source, it will add an entry to a FIFO queue model according to the destination address of this packet. At the same time, the process will start a retransmission timer based on self-interrupts to wait for an acknowledgement from the re-

ceiving node. When an acknowledgement-type packet is received before this timer expires, the corresponding entry for that ACKD-type packet will be erased from the queue. However, if XMIT_EXPRES is triggered, the expired ACKD-type packet will be retransmitted and a new timer will be started.

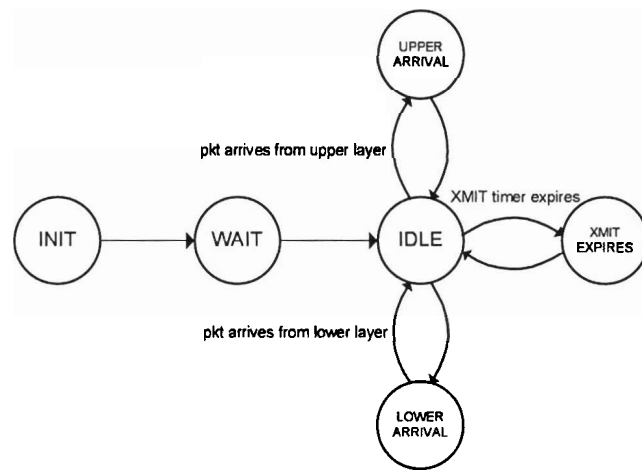


Figure 3.7: Finite state machine of Transport module

3.4.3.4 Application module

The services provided by the application module are: 1) Address assignment, where the addresses of the node and the outgoing messages are specified (or can be set to auto-assign by simulator) 2) Message assignment, where one can specify the message type (e.g. ACKD and UNACKD_RPT), length, and the frequency rates to be generated during the simulation.

The application module makes no assumptions apart from relying on the transaction module for correct TPDU sequencing and duplicate detection of packets. The process model of the application module, which is an extension of the source generator model, is shown in Figure 3.8. The model consists of six states: INIT, WAIT, ACTIVE, GENERATE_PACKET, RECEIVE_PACKET, and STOP.

The packet generation service and receiving process are implemented in the GENER-

ATE_PACKET and RECEIVE_PACKET states, respectively. The STOP state is used to collect statistical values at the end of the simulation.

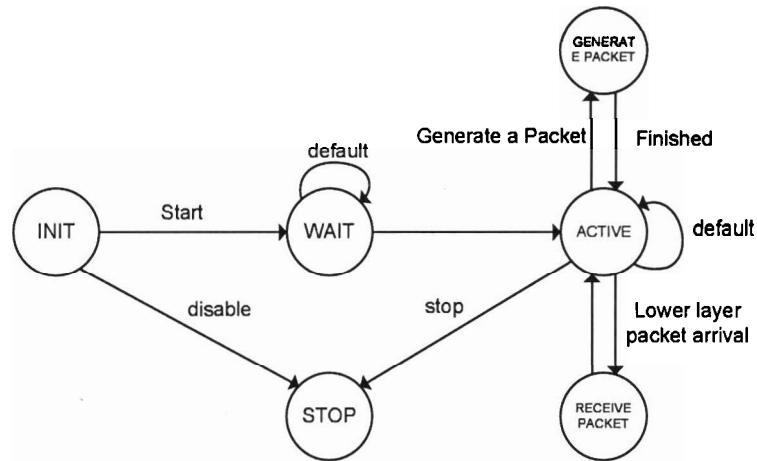


Figure 3.8: Finite state machine of Application module

3.5 Model Validation

We discuss numerical results concerning the performance of the LonWorks modeling tool. In addition to the analysis presented in Section 3.3, simulation results of our communication simulator were compared to the results gathered from a physical test platform. The purpose of the comparison are twofold: 1) to cross-validate the results obtained from two analytical models, and 2) to experiment with variations of the scheme, traffic patterns, and network loads which are not easily represented by the current analysis.

3.5.1 Validation of collision probability

Figure 3.9 illustrates the collision probability of the communication channel as a function of the number of nodes in the channel where the number of slots in the randomization set is a fixed value of 16 slots. The dashed line is the analytical result gathered from Equation 3.5,

and the solid line with star symbols indicates the result gathered from the simulation tool. In each simulation run, every node in the network attempted to transmit repeatedly an unacknowledged floating-point network variable to a neighboring node as fast as possible. The simulated nodes on the channel were set to “5MHz neuron processors” and they all connected to a TP/FT-10 media-type network at a 78.125 kbps data rate.

As expected, the collision probability increases when the number of nodes increases. The OPNET simulation results match the collision rates predicted by Equation 3.5 closely. As we can see from the figure, when the network size and number of collisions increase, the difference between the analytical result and simulation result also increases, since our assumption of a fixed window size W is less valid when there are more nodes competing for the channel. This situation results in an increasing number of collisions and backlog window size. Actually, the increase in the value of W by the protocol allows the physical network to have a better collision rate than the rate our analytical model predicted.

Figure 3.10 shows the probability of unsuccessful transmission for the given time slot when all the nodes connected to the channel were set to *acknowledged* message service. The dashed line represents the analytical result gathered from Equation 3.18 and 3.19. The solid line represents the result gathered from simulation. During the simulation, every node in the network always tried to transmit a packet as long as the node sensed that the channel was idle. We set the parameters of the simulation in accordance with our node-oriented analytical model. The service type of transmission was “acknowledged”, which results in dynamic adjustment of the size of the randomizing window to the current estimated channel backlog S ranging between 1 and 63. The simulated nodes on the channel were set to 5MHz again and a TP/FT-10 media-type network was still used.

From the figure we can see that the simulation results match the trend of the analysis, with greater discrepancy for networks with a small number of nodes connected on it. In these small networks, the underlying assumption of our analysis, namely that every node is ready to transmit whenever the channel is available, does not hold due to hardware limita-

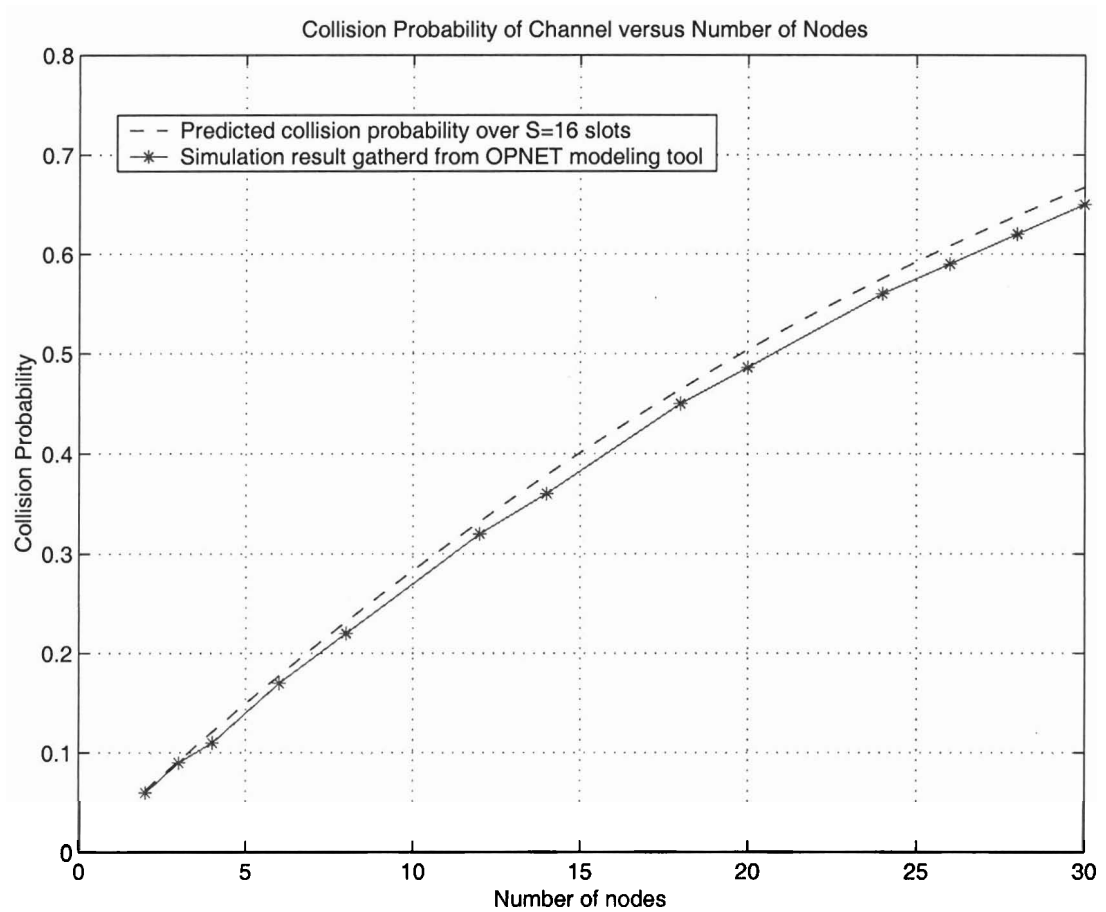


Figure 3.9: Comparison of channel collision probability

tions. As a result, actual collision rates are lower than those predicted by Equation 3.18 and 3.19. For networks with a large number of nodes, the channel is occupied with activity and now the channel, not the local node processors, becomes the limiting factor in the system. Messages in the media access processor begin to queue up, guaranteeing that whenever the channel returned to an idle state, every node in the network would position itself in the transmission queue.

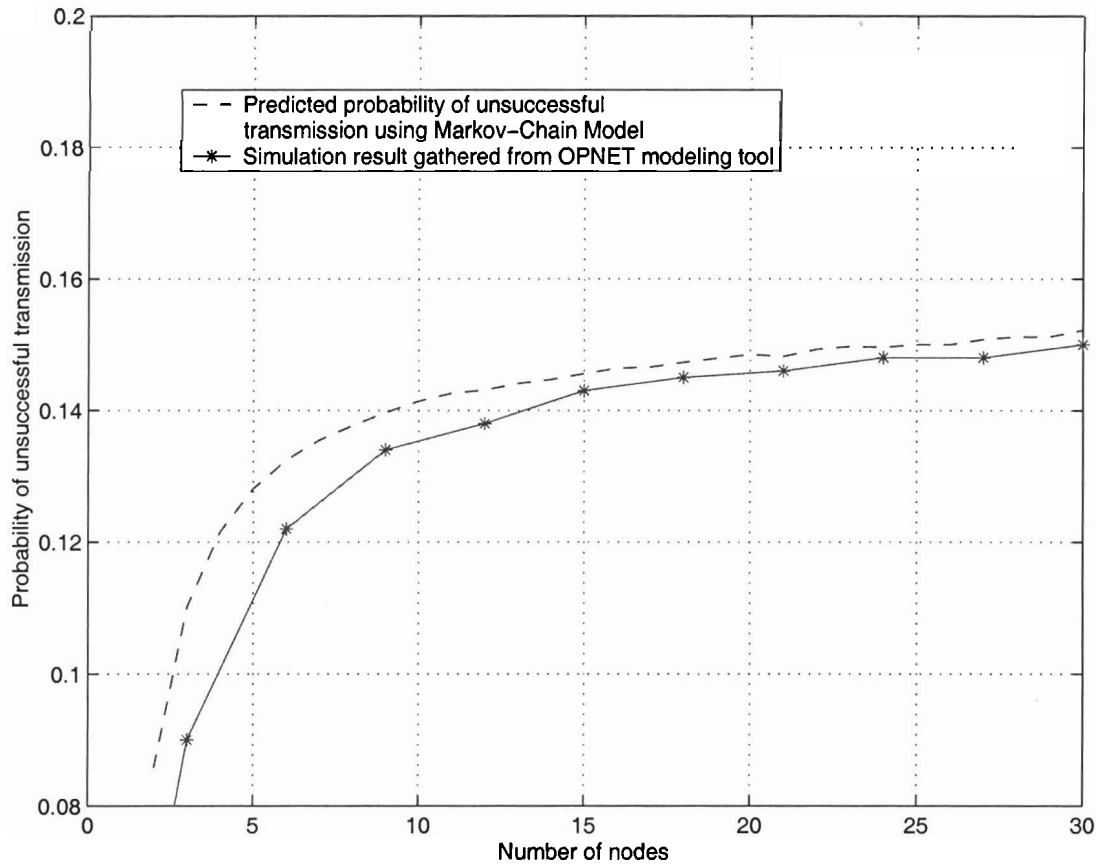


Figure 3.10: Comparison of Node collision probability

3.5.2 Validation of channel throughput and bandwidth utilization

In order to compare the accuracy of our modeling tool, we compared the results of the simulation to the results gathered from a measurement of a physical system based on six (6) programmable smartcontrol devices. We demonstrate two examples in this section. Test 1 uses an *unacknowledged* message service while test 2 uses an *acknowledged* service. In both cases, all the smartcontrol devices were connected with twisted-pair wiring and configured to transmit at 78.125 kbps. All the nodes were programmed to transmit floating-point type network variable to a neighboring node according to the variable rates. We configured the exact same settings in the LonWorks modeling tool.

Figure 3.11 shows the comparison of channel throughput as a function of the traffic

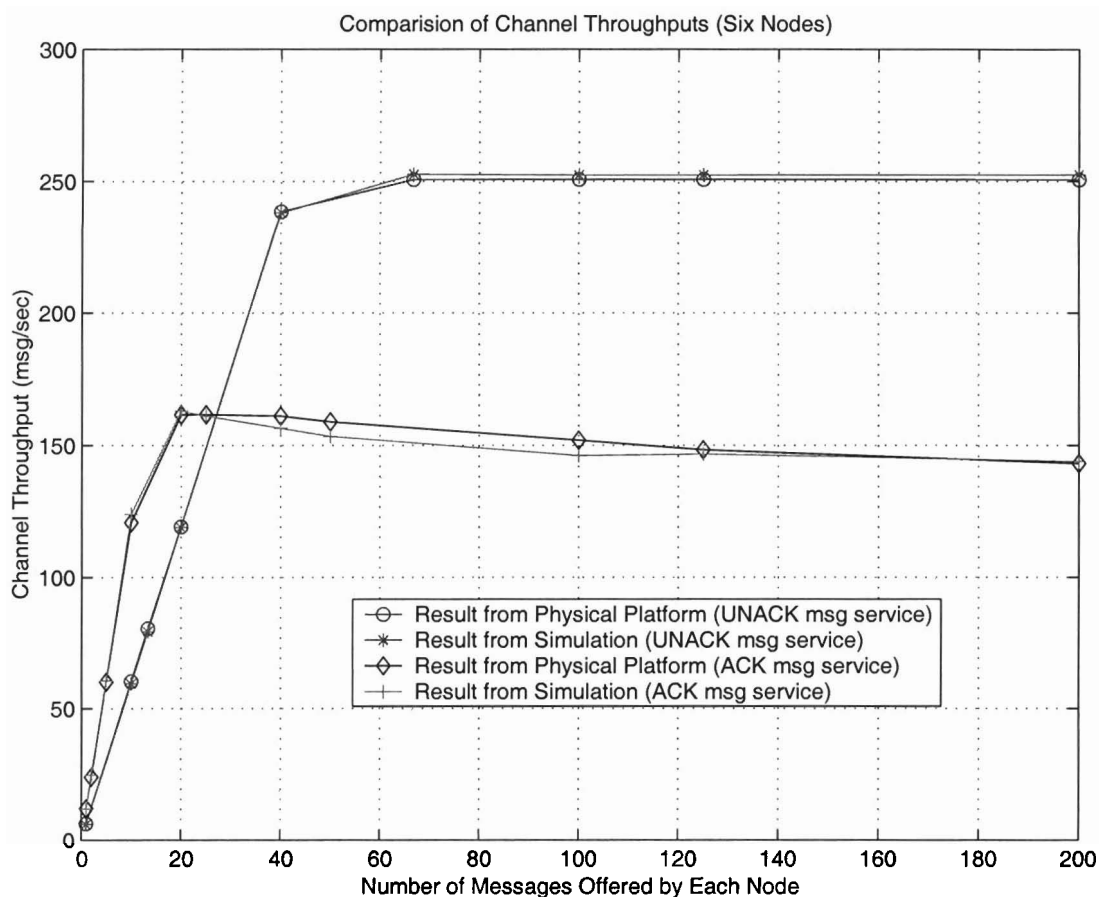


Figure 3.11: Comparison of Channel Throughput

offered by each node for tests 1 and 2. The channel throughput is expressed in messages transmitted per second, which is the actual message load to the network. Figure 3.12 shows the comparison of bandwidth utilization for tests 1 and 2. The bandwidth utilization is defined as the time used for transmission divided by the total elapsed time of the simulation. In both plots, the line crossed by the circle represents data gathered experimentally; the line crossed by the star represents the corresponding throughput rates predicted by our modeling tool. From Figures 3.11 and 3.12, we can see that the simulation model matches the real system well. We note that under a light traffic load, the channel throughput increases linearly until a saturation point.

Chapter 4. An Efficient Method for DOA Estimation: Unitary Improved Polynomial Rooting

4.1 Introduction

The final chapter in this thesis addresses a different aspect of sensor arrays, namely the use of arrays for direction of arrival estimation for target detection and target tracking. Over the last 30 years, this problem, tracking the DOA of multiple targets by using a sensor network, has received considerable attention. Many techniques have been developed for important applications such as sonar, radar, radio, telecommunication, astronomy and strategic defense operations [12, 16, 47]. For example, in wireless communication (W-CDMA, GSM) [22, 31], a smart antenna system uses an array of sensors to acquire the spatial signatures of the transmitted signals. It uses the difference of the spatial signatures or the DOA of signals in order to estimate the desired signal. The array sensors may consist of antennas as in radar, radio communication [25] and radio astronomy, hydrophones as in sonar applications, geophones in seismology and ultrasonic probes and X-ray detectors in medical imaging [21]. The objective is to extract information such as estimates of number of signals, direction of arrival (DOA), and central frequency from the sensors' output. Since the collected data contain noise, signal processing is required for the suppression of noise and interfering signals and the enhancement of the target signals.

Several high resolution methods for DOA estimation such as MUSIC [52], Root-MUSIC [10], *Minimal Norm* (MIN-NORM) [29], *Estimation of Signal Invariance Techniques* (ESPRIT) [51] and the IPR method [9] have been developed. Those methods based on complex value signal subspace often require significant computational cost during eigen-decomposition. Other methods based on maximum likelihood principles are often dependent on assumptions and have a narrow scope of applicability. Recently, several techniques have concentrated on reducing the computational complexity of eigen-decomposition based methods: Pesavento

and Haardt [44] presented a real-valued variant of Root-MUSIC based on the unitary transformation. Lineborge [35] developed several efficient algorithms for real value singular value decomposition (SVD).

In this chapter, a unitary transformation of the IPR method [32] is developed. The method transforms a complex-valued covariance matrix into a real-valued matrix by unitary transformations, and the IPR method is used to determine the DOA. The use of Unitary-IPR reduces the computational cost by more than half compared to IPR, while DOA precision is maintained.

The rest of this chapter is organized as follows: we discuss the array signal model in Section 4.2. In Section 4.3, we describe two existing techniques: MUSIC and root-MUSIC, then we propose Unitary-IPR method in Section 4.4. After that, we present the simulation results of the proposed method and compare it to other existing methods. In Section 4.5 we show the simulation results. Finally, we conclude that our method achieve the better results from the stances of computational complexity and mean square error of DOA estimation.

4.2 Model Definition and Problem Formulation

Consider q signals represented as narrow-band processes, emitted by the targets from the far field. The signals can be considered as sample functions of a stationary stochastic process or deterministic functions in time. They are impinging on an uniform linear array (ULA) composed of p omnidirectional isotropic sensors. The signals are assumed point sources from q targets. The signals have a known identical center frequency ω_0 , and emit in (azimuth) directions $\theta_1, \theta_2, \dots, \theta_q$ where $-\frac{\pi}{2} \leq \theta_i \leq \frac{\pi}{2}$.

The k th sensor observation, $y_k(t)$, consists sum of all point sources $s_i(t)$ ($i = 1, \dots, q$). Also, $y_k(t)$ includes additive noise, $\eta_k(t)$, assumed white Gaussian noise with zero mean and uncorrelated with $s(t)$. The measurement of the k th sensor $y_k(t)$ is expressed as:

$$y_k(t) = \sum_{i=1}^q a_k(\theta_i) s_i(t) + \eta_k(t), \quad (4.1)$$

where $k = 1, 2, \dots, p$, and the a_k is a steering factor that $a_k(\theta_i) = e^{-j\omega_0\tau_k(\theta_i)}$. $a_k(\theta_i)$ is specified by propagation time delay $\tau_k(\theta_i)$ at k th sensor by i th signal, and it assumed to be $(d_k/c)\sin(\theta_i)$, where d_k is the distance between the k th sensor and the reference point of the first sensor, c is the speed of wave propagation, and θ_i is the DOA for the i th point source.

We can rewrite equation (4.1) in matrix form as

$$\mathbf{Y}(t) = \mathbf{A}(\boldsymbol{\theta})\mathbf{S}(t) + \boldsymbol{\eta}(t), \quad (4.2)$$

where

$$\begin{aligned} \mathbf{Y}(t) &= [y_1(t), y_2(t), \dots, y_p(t)]^T, \\ \mathbf{S}(t) &= [s_1(t), \dots, s_q(t)]^T, \\ \mathbf{A}(\boldsymbol{\theta}) &= [\mathbf{a}(\theta_1), \dots, \mathbf{a}(\theta_q)], \\ \mathbf{a}(\theta_i) &= [e^{-j\omega_0\tau_1(\theta_i)}, e^{-j\omega_0\tau_2(\theta_i)}, \dots, e^{-j\omega_0\tau_p(\theta_i)}]^T, \\ \boldsymbol{\eta}(t) &= [\eta_1(t), \eta_2(t), \dots, \eta_p(t)]^T. \end{aligned}$$

When the sensors are located equally spaced, we have

$$\mathbf{a}(\theta) = [1 \ e^{-j(\frac{\omega_0 d}{c})\sin\theta} \dots e^{-j(\frac{\omega_0(p-1)d}{c})\sin\theta}]^T \quad (4.3)$$

where d is the distance between two adjacent sensors. The array is said to be unambiguous if the corresponding vectors $\mathbf{a}(\theta_i)$ for distinct θ_i are linearly independent or, equivalently, \mathbf{A} is of full rank.

Assume that the signal $\mathbf{S}(t)$ ($q \times 1$) has mean zero and a non-singular covariance matrix:

$$\boldsymbol{\Gamma}_s = E(\mathbf{S}(t)\mathbf{S}^\dagger(t)), \quad (4.4)$$

where $\mathbf{S}^\dagger(t)$ denotes the complex conjugate and transpose of the matrix $\mathbf{S}(t)$, and that the noise $\boldsymbol{\eta}(t)$ ($p \times 1$) is white and has zero mean and covariance matrix $\boldsymbol{\Gamma}_n = E(\boldsymbol{\eta}(t)\boldsymbol{\eta}^\dagger(t)) = \sigma^2\mathbf{I}_p$, where \mathbf{I}_p is $p \times p$ identity matrix.

Furthermore, $\eta(t)$ is assumed to be independent of $S(t)$. The covariance matrix of $\mathbf{Y}(t)$ is

$$\Sigma = E(\mathbf{Y}(t)\mathbf{Y}^\dagger(t)) \quad (4.5)$$

$$= \mathbf{A}\Gamma_s\mathbf{A}^\dagger + \sigma^2\mathbf{I}_p. \quad (4.6)$$

It is common to obtain the correlation matrix estimate $\hat{\Sigma}$ based on N snapshots using

$$\hat{\Sigma} = \frac{1}{N} \sum_{t=1}^N \mathbf{Y}(t)\mathbf{Y}^\dagger(t) \quad (4.7)$$

By *spectral decomposition* (SD), there exists an orthogonal matrix \mathbf{E} of order p and a real diagonal matrix Λ of order p such that

$$\begin{aligned} \Sigma &= \mathbf{A}\Gamma_s\mathbf{A}^\dagger + \sigma^2\mathbf{I}_p \\ &= \mathbf{E}\Lambda\mathbf{E}^\dagger, \end{aligned} \quad (4.8)$$

where

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p), \quad (4.9)$$

and

$$\mathbf{E} = [\mathbf{e}_1 \cdots \mathbf{e}_q; \mathbf{e}_{q+1} \cdots \mathbf{e}_p] \quad (4.10)$$

In this thesis, lowercase boldface characters refer to vectors. Uppercase boldface italic characters refer to matrices. A^\dagger is the Hermitian conjugate (or complex-conjugate transpose) of matrix A . Subspace methods use the special eigenstructure of Σ which is expressed in terms of its eigenvalues, λ_n , and their corresponding eigenvectors \mathbf{e}_n ($n = 1, 2, \dots, p$). We $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$. The first q eigenvalues corresponding to the signal should be larger than σ^2 , and the remaining $(p - q)$ eigenvalues are approximately

equal to σ^2 . The eigenvectors corresponding to the signal eigenvalues can be described using the signal subspace, $E_s = [e_1 \dots e_q]$. $E_n = [e_{q+1} \dots e_p]$ is the matrix containing the remaining $p - q$ noise eigenvectors describing the noise subspace, which is the orthogonal complement to the signal subspace.

4.3 Improved Polynomial Rooting

Proposed by Bai, Improved Polynomial Rooting method (IPR) is an efficient and asymptotic method to estimate the DOA.

If we let $z_l = e^{j(\omega_0 d/c) \sin \theta_k}$ ($k = 1, 2, \dots, q$) and define a polynomial as,

$$g_b(z) = g_{q+1}z^q + \dots + g_1 \quad (4.11)$$

$$= g_{q+1} \prod_{i=1}^q (1 - z_l z^{-1}) = 0. \quad (4.12)$$

Suitably choose g_{q+1} so that we have

$$\sum_{i=1}^{q+1} |g_i|^2 = 1, \quad (4.13)$$

and $g_{q+1} > 0$. Then, we have

$$\mathbf{a}^\dagger(\theta_k) \mathbf{G} = 0, \quad k = 1, \dots, q.$$

Where G is $p \times (p - q)$ matrix given by

$$\mathbf{G} = \begin{pmatrix} g_{1,q+1} & 0 & \dots & 0 \\ g_{2,q+1} & g_{2,q+2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ g_{q+1,q+1} & g_{q+1,q+2} & \ddots & \hat{g}_{q+1,p} \\ 0 & \hat{g}_{q+2,q+2} & \ddots & g_{q+2,p} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & g_{p,p} \end{pmatrix}_{p \times (p-q)}. \quad (4.14)$$

the estimated noise eigen vectors matrix \hat{U}_n can always be transformed to \hat{G} by Gauss elimination or Householder transformation. The coefficients of $g(z)$ can be estimated by averaging the elements of the $p - q$ columns of \hat{G} :

$$\hat{g}_i = \frac{1}{p - q} (\hat{g}_{i,q+1} + \hat{g}_{i+1,q+2} + \cdots + \hat{g}_{i+p-q-1,p}) \quad (4.15)$$

for $i = 1, \dots, q + 1$. As a consequence, the exactly q roots of the polynomial equation

$$\hat{g}(z) = \sum_{i=1}^{q+1} \hat{g}_i z^{i-1} = 0$$

with the form

$$\hat{z}_k = \hat{\rho}_k e^{-j\hat{\tau}_k}, \quad k = 1, \dots, q. \quad (4.16)$$

will converge to $z_k = e^{-\tau_k}$ with probability 1. Thus, the DOAs corresponding to the array A can be obtained by $\hat{\tau}_k$.

4.4 Proposed Method– Unitary-IPR

In this section, a unitary transformation of the IPR method is proposed. The covariance matrix estimate in equation(4.7) can be described as forward averaged matrix [45], since the matrix can be found by a preprocessing scheme that partitions the total array of sensors into subarrays and then generates the average of the subarray output covariance matrix. Usually, we call matrix Σ as centro-Hermitian [23] if there exists a matrix J such that

$$\Sigma = J \Sigma^* J \quad (4.17)$$

where J is the exchange matrix with ones on its anti-diagonal and zeros elsewhere. Σ^* represents Σ conjugate. Σ is the centro-Hermitian only when the signal sources are uncorrelated. For the purpose of decorrelating any correlated signal matrix S that we previously defined, a forward-backward (FB) matrix [35] has been developed to make the signal co-

variance matrix Σ become centro-Hermitian matrix Σ_{FB} as

$$\begin{aligned}\widehat{\Sigma}_{FB} &= \frac{1}{2}(\widehat{\Sigma} + \mathbf{J}\widehat{\Sigma}^* \mathbf{J}) \\ &= \mathbf{A}\tilde{\Gamma}\mathbf{A}^\dagger + \sigma^2 \mathbf{I},\end{aligned}\quad (4.18)$$

where

$$\tilde{\Gamma} = \frac{1}{2}(\Gamma + \mathbf{D}\Gamma^* \mathbf{D}^\dagger), \quad (4.19)$$

and

$$\mathbf{D} = \text{diag}\{e^{-j(\frac{w_0 d(p-1)}{c}) \sin \theta_1}, \dots, e^{-j(\frac{w_0 d(p-1)}{c}) \sin \theta_q}\} \quad (4.20)$$

The centro-hermitian matrices can be interpreted to real-valued (unitary) matrices [35] by

$$\widehat{\mathbf{L}} = \mathbf{K}^\dagger \widehat{\Sigma}_{FB} \mathbf{K}, \quad (4.21)$$

where $\widehat{\mathbf{L}}$ is $q \times q$ square matrix, \mathbf{K} is the matrix that satisfies when Σ_{FB} is even dimension,

$$\mathbf{K} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & j\mathbf{I} \\ \mathbf{J} & -j\mathbf{J} \end{pmatrix} \quad (4.22)$$

and when Σ_{FB} is odd dimension, we have

$$\mathbf{K} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbf{I} & 0 & j\mathbf{I} \\ 0 & \sqrt{2} & 0 \\ \mathbf{J} & 0 & -j\mathbf{J} \end{pmatrix} \quad (4.23)$$

Now we can get the real-valued matrix $\widehat{\mathbf{L}}$ directly via the complex valued covariance matrix $\widehat{\Sigma}$ since

$$\begin{aligned}\widehat{\mathbf{L}} &= \mathbf{K}^\dagger \widehat{\Sigma}_{FB} \mathbf{K} \\ &= \frac{1}{2}(\mathbf{K}^\dagger \widehat{\Sigma} \mathbf{K} + (\mathbf{K}^*)^\dagger \widehat{\Sigma}^* \mathbf{K}^*) \\ &= \text{Real}\{\mathbf{K}^\dagger \widehat{\Sigma} \mathbf{K}\}.\end{aligned}\quad (4.24)$$

\widehat{L} can be obtained via either $\widehat{\Sigma}_{FB}$ or $\widehat{\Sigma}$, the two approaches are equivalent, however, the latter has a simpler implementation than the former.

Consider the real-valued eigen-decomposition of L is

$$\begin{aligned} L &= \mathbf{V}\mathbf{\Pi}\mathbf{V}^\dagger \\ &= \mathbf{V}_s\mathbf{\Pi}_s\mathbf{V}_s^\dagger + \sigma^2\mathbf{V}_n\mathbf{V}_n^\dagger, \end{aligned} \quad (4.25)$$

where $\mathbf{\Pi}$ is eigenvalue matrix and \mathbf{V} is the eigenvector matrix of L .

$$\mathbf{\Pi}_s = \text{diag}[\pi_1, \dots, \pi_q]$$

$$\mathbf{V}_s = [v_1, \dots, v_q]$$

$$\mathbf{V}_n = [v_{q+1}, \dots, v_p]$$

We also have the eigen-decomposition of the matrix Σ , which has been given in equation (4.8).

$$\begin{aligned} \Sigma &= \mathbf{E}\mathbf{\Lambda}\mathbf{E}^\dagger \\ &= \mathbf{E}_s\mathbf{\Lambda}_s\mathbf{E}_s^\dagger + \sigma^2\mathbf{E}_n\mathbf{E}_n^\dagger \end{aligned} \quad (4.26)$$

where

$$\mathbf{\Lambda}_s = \text{diag}\{\lambda_1, \dots, \lambda_q\}$$

$$\mathbf{E}_s = [e_1, \dots, e_q]$$

$$\mathbf{E}_n = [e_{q+1}, \dots, e_p]$$

The FB matrix of Σ , Σ_{FB} is defined in the same way

$$\Sigma_{FB} = \mathbf{U}\mathbf{\Xi}\mathbf{U}^\dagger \quad (4.27)$$

$$= \mathbf{U}_s\mathbf{\Xi}_s\mathbf{U}_s^\dagger + \sigma^2\mathbf{U}_n\mathbf{U}_n^\dagger \quad (4.28)$$

The eigenvalues and eigenvectors are defined as follows,

$$\begin{aligned}\Xi_s &= \text{diag}\{\xi_1, \dots, \xi_q\} \\ U_s &= [u_1, \dots, u_q] \\ U_n &= [u_{q+1}, \dots, u_p]\end{aligned}$$

Let us rewrite the equation (4.27),

$$\Sigma_{FB}U = U\Xi \quad (4.29)$$

and

$$\begin{aligned}K^\dagger \Sigma_{FB}U &= K^\dagger \Sigma_{FB}KK^\dagger U \\ &= LK^\dagger U\end{aligned} \quad (4.30)$$

$$= K^\dagger U\Xi \quad (4.31)$$

From equations (4.30) and (4.31), the eigenvectors and the eigenvalues of the matrix L and Σ_{FB} are coupled as

$$U = KV \quad (4.32)$$

$$\Xi = \Pi \quad (4.33)$$

The noise eigenvectors of U_n and V_n should also be hold as,

$$U_n = KV_n \quad (4.34)$$

Equation (4.34) indicates that real-value eigen-decomposition can be used to find the eigenvectors of the signal covariance matrix. Also one can notice that the floating point operation cost for directly evaluating the eigenvectors and eigenvalues of Σ_{FB} is in the order of $8n^2$, which are, $4n^2$ for multiplications and another $4n^2$ for additions. If we use the real-value eigen decomposition (4.25), it only uses n^2 for multiplications and n^2 for

$$\begin{aligned}\Xi_s &= \text{diag}\{\xi_1, \dots, \xi_q\} \\ U_s &= [u_1, \dots, u_q] \\ U_n &= [u_{q+1}, \dots, u_p]\end{aligned}$$

Let us rewrite the equation (4.27),

$$\Sigma_{FB}U = U\Xi \quad (4.29)$$

and

$$\begin{aligned}K^\dagger \Sigma_{FB}U &= K^\dagger \Sigma_{FB}K K^\dagger U \\ &= LK^\dagger U\end{aligned} \quad (4.30)$$

$$= K^\dagger U\Xi \quad (4.31)$$

From equations (4.30) and (4.31), the eigenvectors and the eigenvalues of the matrix L and Σ_{FB} are coupled as

$$U = KV \quad (4.32)$$

$$\Xi = \Pi \quad (4.33)$$

The noise eigenvectors of U_n and V_n should also be hold as,

$$U_n = KV_n \quad (4.34)$$

Equation (4.34) indicates that real-value eigen-decomposition can be used to find the eigenvectors of the signal covariance matrix. Also one can notice that the floating point operation cost for directly evaluating the eigenvectors and eigenvalues of Σ_{FB} is in the order of $8n^2$, which are, $4n^2$ for multiplications and another $4n^2$ for additions. If we use the real-value eigen decomposition (4.25), it only uses n^2 for multiplications and n^2 for

additions, total is $2n^2$. That is 75% of the computational cost efficiency compared to direct evaluation.

The Unitary-IPR algorithm is described as follows,

1. Determine covariance matrixes $\hat{\Sigma}$ from the sensors' observation,
2. Obtain a real-value matrix \bar{L} by Equation (4.24),
3. Solve eigenvalue matrix $\Pi = \text{diag}(\pi_1, \dots, \pi_p)$ and eigenvector matrix $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p]$ from \hat{L} ,
4. Obtain noise eigen-vector matrix $\hat{\mathbf{V}}_n = [\mathbf{v}_1 \cdots \mathbf{v}_{q+1}]$ from \mathbf{V} ,
5. Convert to U_n by Equation (4.34),
6. Column reduce \hat{U}_n into G_u ,
7. Sum the values in each column of \hat{G}_u to obtain the value of \hat{g}_i ,
8. Solve the roots \hat{z}_k from the polynomial equation with coefficients \hat{g}_i , where $1 \leq i \leq q + 1$ and $1 \leq k \leq q$, and
9. Determine the DOAs $\theta_k = \arcsin(\text{Real}(-\frac{c \ln(\hat{z}_k)}{i\omega_0 d}))$.

4.5 Simulation Results

In this section, we study the performance of the proposed Unitary-IPR method using numerical simulations. We compared our method with the existed methods we have mentioned in Section 4.1, namely MUSIC, Root-MUSIC, TLS-ESPRIT and IPR.

A uniformly linear array of p sensors is used with inter-element spacings of d . Suppose the central frequency ω_0 of the targets is 10^8 Hz.

4.5.1 Comparisons of RMSE and FPOC vs SNR

Assume we have three uncorrelated targets of equal power located at $\theta_1 = -20^\circ, \theta_2 = 10^\circ$ and $\theta_3 = 40^\circ$. Let $d = 10$ and $p = 10$. We take 200 snapshots with 1000 times simulation. We can show the accuracy and the computational complexity of the DOA estimation. The accuracy is determined by *Root Mean Square Error* (RMSE), and computational complexity is measured by number of Floating Point Operation Counts(FPOCs).

The RMSE is plotted vs. SNR (-5dB–15dB) in figure 4.1. The RMSE estimator shows a significant improvement compared to IPR and TLS-ESPRIT.

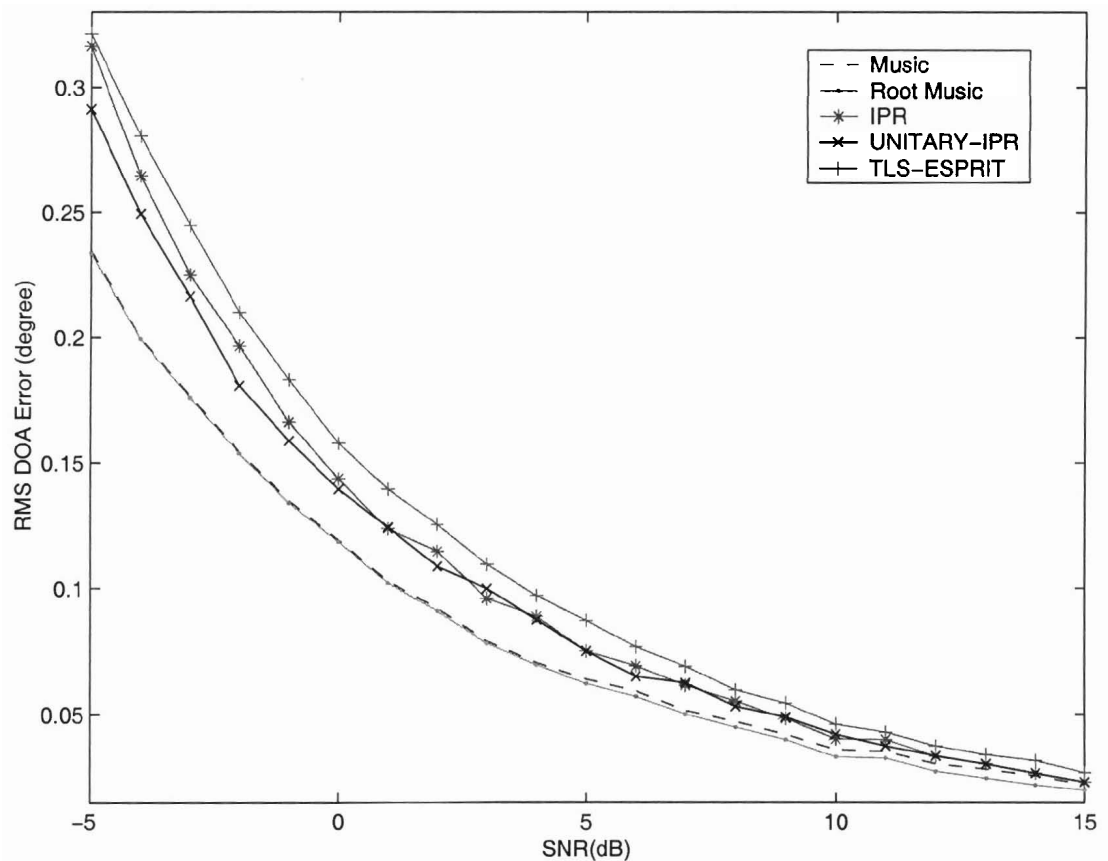


Figure 4.1: The RMSE vs. SNR performance of Unitary-IPR compared to MUSIC, root-MUSIC, TLS-ESPRIT, and IPR($q = 3, p = 10$, and 200 snapshots)

The computation complexity can be evaluated using flops function in matlab. Figure 4.2

illustrates the FPOCs of each methods vs. SNR. Again, Unitary-IPR is much better compared to other methods because it requires the least FPOCs.

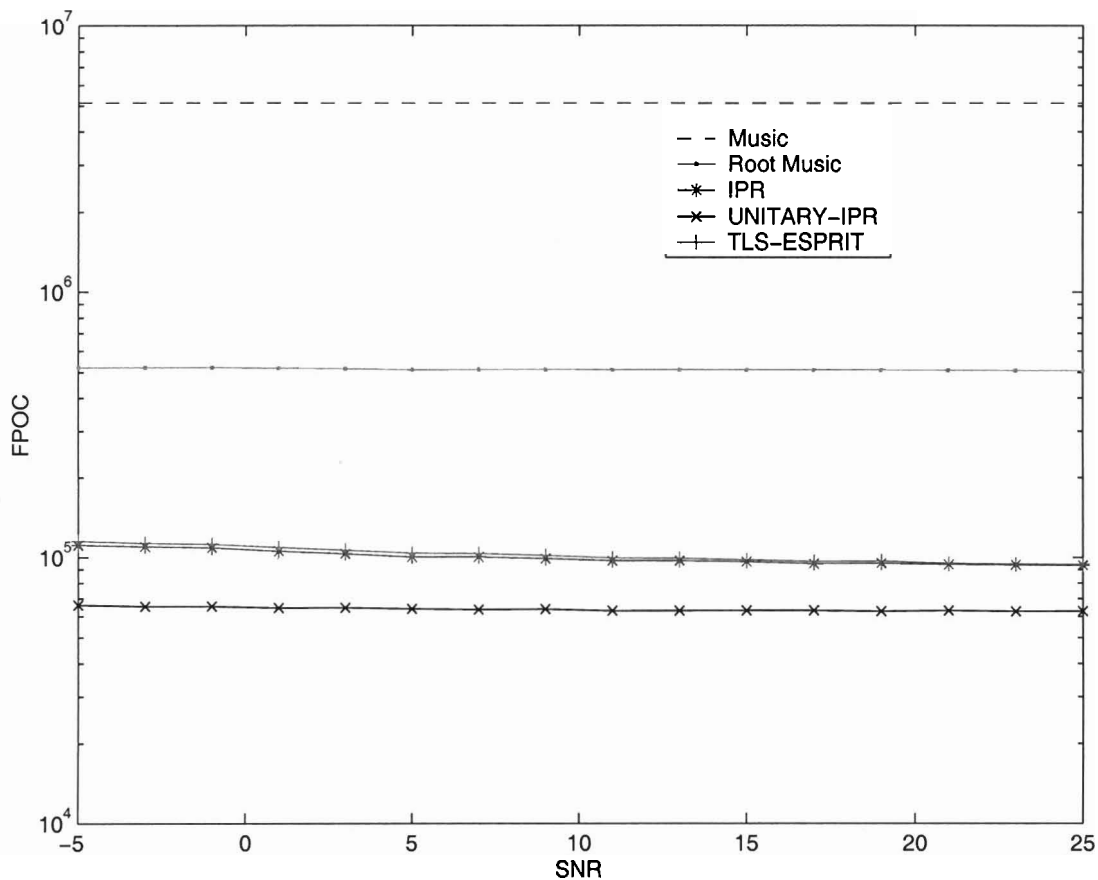


Figure 4.2: The Floating Point Operation Counts vs. SNR performance of Unitary-IPR compared to other methods($q = 3$, $p = 10$, and 200 snapshots)

4.5.2 Comparisons of RMSE and FPOC vs number of sensor

Now we can study the performance measures when we vary number of sensors while we keep SNR equals to 5dB. The other parameters stay the same as in the Section 4.5-A. Figure 4.3 shows the RMSE comparison, and figure 4.4 illustrates the FPOCs comparison with number of sensors varies from 5-25. Again, Unitary-IPR shows a significant improvement over other methods. Also, we observe the FPOC increases exponentially as the number of

sensor increases.

4.5.3 Comparisons of RMSE vs number of snapshots

Here, we assume the SNR is the fixed value at 5dB, and show the RMSE performance with different sample number (snapshots) N . Again, Unitary-IPR achieves a better performance than TLS-ESPRIT and IPR, and the FPOC performance for Unitary-IPR can be figured out from Figure 4.2 at point SNR=5 dB.

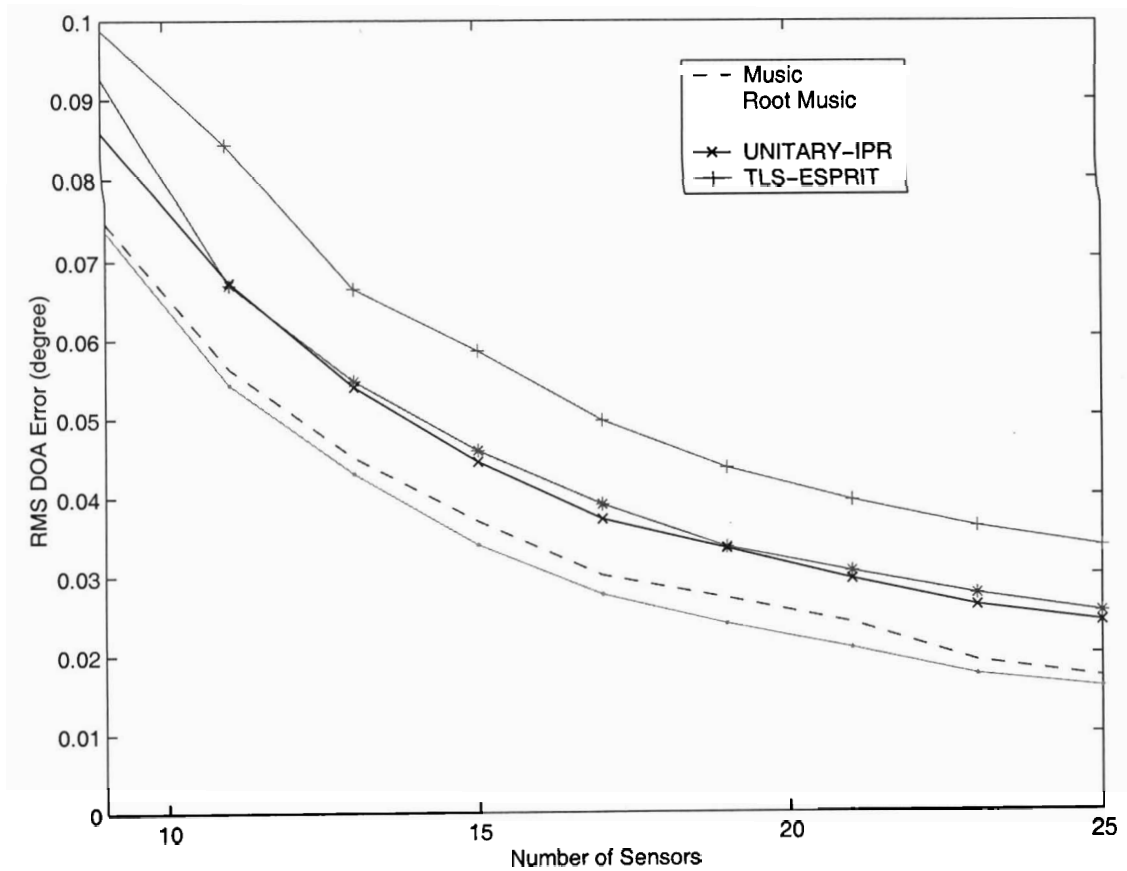


Figure 4.3: Comparison of RMSE vs. Number of Sensors ($q=3$, SNR=5dB, 200 snapshots)

sensor increases.

4.5.3 Comparisons of RMSE vs number of snapshots

Here, we assume the SNR is the fixed value at 5dB, and show the RMSE performance with different sample number (snapshots) N . Again, Unitary-IPR achieves a better performance than TLS-ESPRIT and IPR, and the FPOC performance for Unitary-IPR can be figured out from Figure 4.2 at point SNR=5 dB.

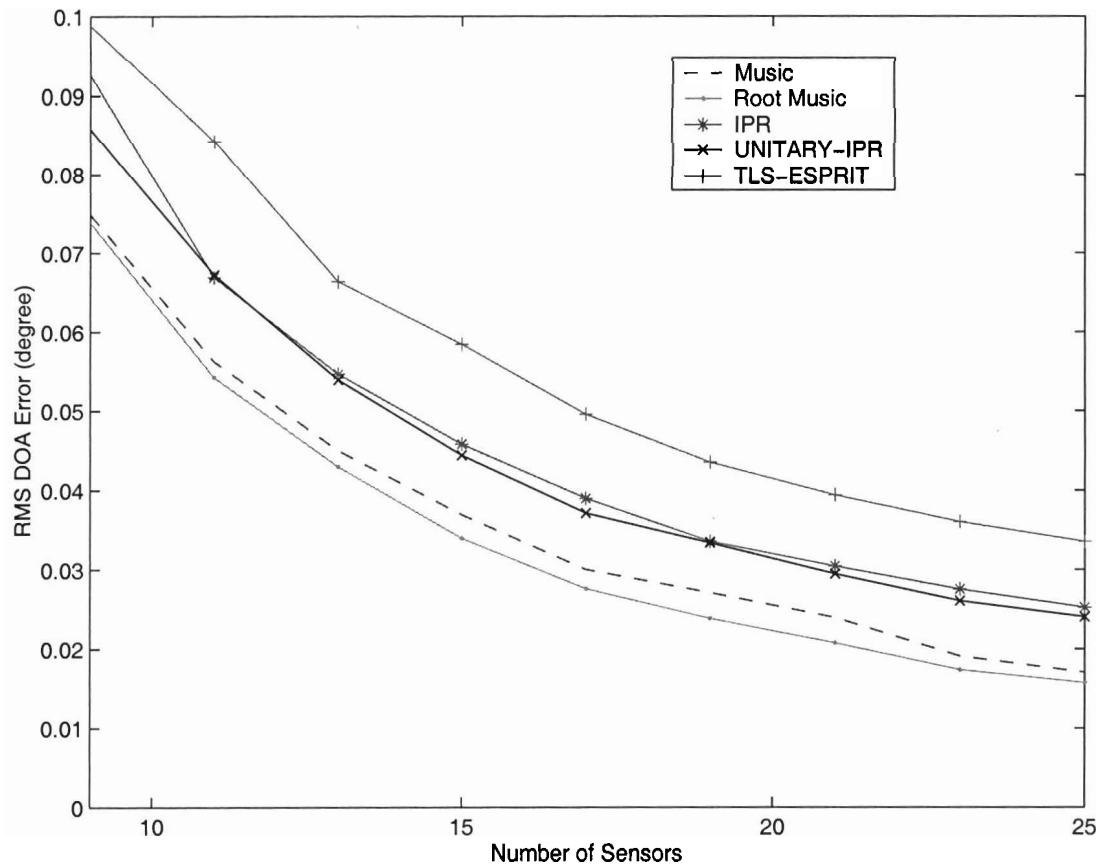


Figure 4.3: Comparison of RMSE vs. Number of Sensors ($q=3$, SNR=5dB, 200 snapshots)

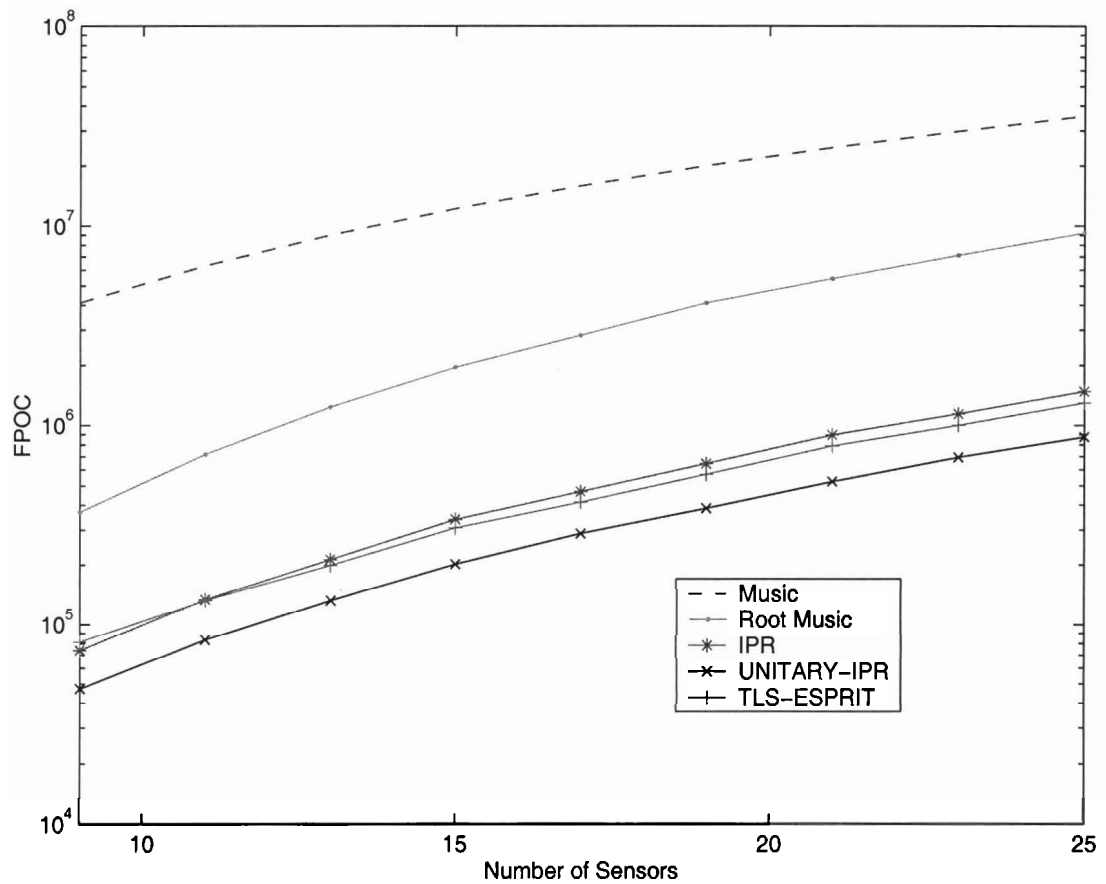


Figure 4.4: Comparison of FPOC vs. Number of Sensors($q=3$, $SNR=5dB$, 200 snapshots)

4.6 Conclusion

We have developed the Unitary Improved Polynomial Rooting method (Unitary-IPR). The simulation results show that it significantly improve the existed IPR method.

Further improvement might be achieved using spatial downsampling of the sensor arrays.

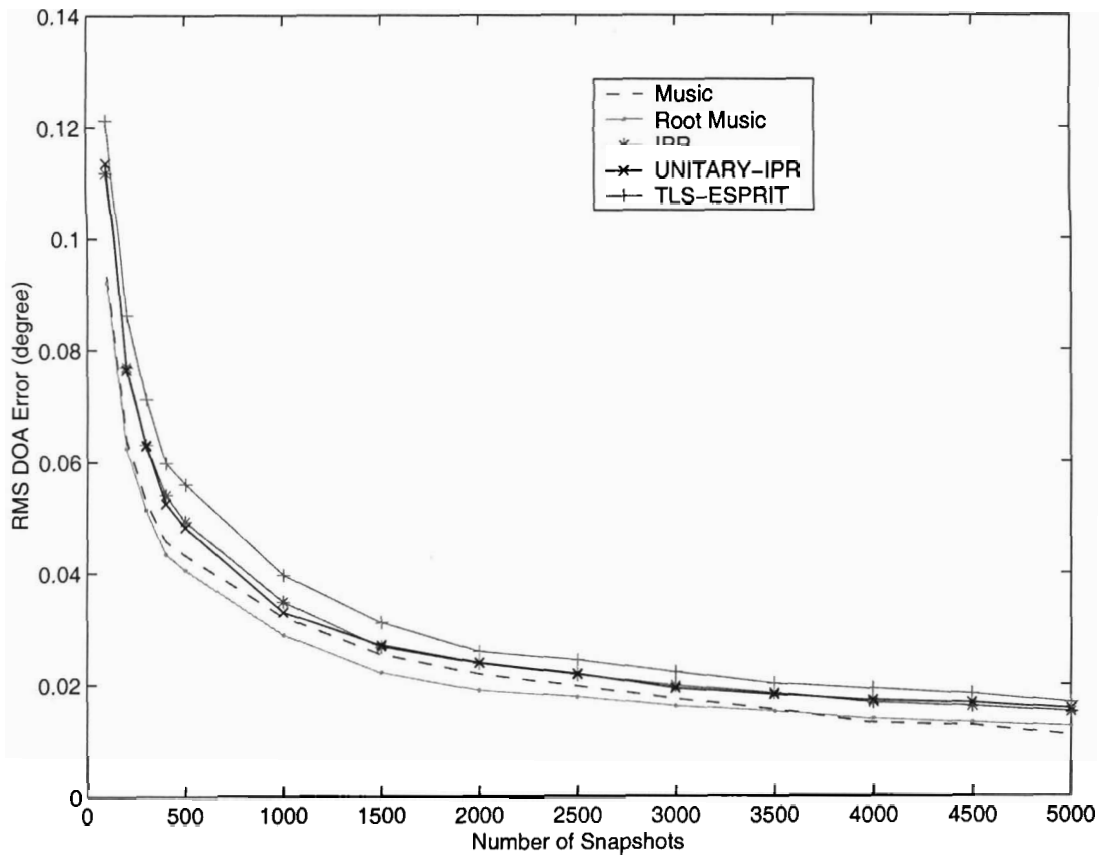


Figure 4.5: Comparison of RMSE vs. Number of Snapshots ($q=3$, $p=10$, $\text{SNR}=5\text{dB}$)

Chapter 5. Conclusion and Future Research

5.1 Technical Summary

In this thesis, we studied distributed sensor network architectures and their applications to the industrial control protocol LonWorks. We have also investigated DOA estimation by sensor networks and improved a popular DOA estimation algorithm by using unitary transformations.

The following results were presented.

- A self-adaptive collision resolution algorithm for the local sensor has been developed. The performance of a distributed detection system employed with such sensor model has been studied.
- A mathematical model for Lonworks control network was developed, to study the network behavior of predictive p -CSMA algorithm.
- A simulator for Lonworks based on OPNET has been implemented.
- A more efficient technique has been developed to estimate the DOA by using an array of sensors.

5.2 Future Research

Future research in directions outlined by this thesis should focus on developing more realistic data fusion solutions. Specifically, the following problems need to be addressed.

An efficient transmission scheme. In our sensor model for distributed sensor networks, we assumed all local sensors have the same detection performance (e.g., all P_f and P_d are same), though in realistic scenarios it is possible that some sensors will have better operating characteristics than others. A more advanced transmission scheme can

be developed so that a priority phase will be introduced. Sensors with higher operating characteristics will be given priorities to improve overall system performance.

Extend distributed detection algorithm when a fading and noisy channel is considered. Fusion of binary decisions transmitted over fading channels has important applications in low-cost low-power wireless sensor networks. The work reported here was based on noise-free environment, with complete channel knowledge. Decision fusion schemes are of interests for single random access channel when fading statistics are available or need to be estimated. Further, the dual problem to decision fusion, namely the optimal local sensor decision rule in the presence of unideal transmission channels should also be investigated.

Appendix A. LonWorks® Analysis Toolkit OPNET Model User's Guide

A.1 Overview

A.1.1 Introduction of LonWorks OPNET Model

The LonWorks OPNET model is a design toolkit that can be used as a simulation, network analysis, testing, and learning tool for LonWorks products. It assists a designer who used LonMaker in the following tasks: (1) evaluating the performance of the communication infrastructure; (2) displaying simulation results; (3) predicting the behavior of communication channel; and (4) estimating/monitoring the network statistics (*e.g.*, channel throughput and bandwidth utilization). These activities do not require any physical network infrastructure and are performed through computation and simulation.

The LonWorks OPNET model is designed to be intuitive and user friendly. The model package can be shared across hardware and platforms transparently without modification.

Figure A.1 shows the GUI of the LonWorks modeling tool. In the figure, we show eight (8) *smartcontrol*® devices along with a Protocol analyzer (LonManager) connected to the TP/FT-10 bus channel. Users can drag icons from the **Object Palette** window in order to add more *smartcontrol*® devices to the channel. The user can run the simulation by clicking **run simulation** under the **Simulation** popup menu.

A.1.2 Features supported with this release (notation follows standard ANSI/EIA-709.1-A)

The features supported is shown in Table A.1. The LonWorks OPNET model assumes that no propagation delays occur on the communication channel; only transmission delays are considered. Moreover, all stations are deferred for integral slot times of β_2 . Therefore, collisions can occur only at the beginning of the transmission cycle. We assume that packet error is caused only by collisions, and that packet loss is a result of the buffer between the

Table A.1: Features supported with this release

Access Mechanism	Carrier senses multiple access with collision avoidance (CSMA/CA) and optional collision detection (CSMA/CD) access schemes as defined in the standard.
Message Service	We support two kinds of message transmissions: acknowledged messages and unacknowledged messages.
Deference & Backoff	Interframe spacing β_1 and backoff time slot β_2 implementation. The values are selected based on design specification from Echelon (the manufacturer of LONWORKS nodes). Random delay backlog generated from the predictive estimation of channel load BL.
Data Rate	78Kbps data rate supported by LonWorks TP/FT-10 bus channel 1.25Mbps data rate supported by TP/XF-1250 bus channel.
Input Clock	Different input clock frequencies of LonWorks node: 0.625MHz, 1.25MHz, 2.5MHz, 5MHz, 10MHz. Different clocks have different limitations on packet rate and time slot duration.
Transmission	Full-duplex transmissions. FIFO processing of transmission requests. Transmission attempt limit of 256 after collision.
Buffer Size	A packet that has arrived from a higher layer to the LonTalk MAC layer is stored in an output buffer. The buffer size is limited to the adjusted maximum value. Higher-layer packets are dropped once the maximum buffer size is reached.
Framing & Disassembly	Assemble MAC frame before transmission, and disassemble frame when receiving from PHY layer. Frame format is defined as MPDU format specified in the standard, with additional address and message type information in order to evaluate channel performance under different message service types.
Frame Size	Frame sizes, based on measurements from real LonWorks channel, are in the range of 10 to 16 bytes.
Node Auto addressing	All the nodes can be configured for node ID auto-addressing. User can also specify node addresses. However, no subnet or group address is supported yet. Destination address of each packet can be chosen automatically from address pool of all the nodes.
Recovery Mechanisms	Retransmission mechanism for data frame based on failure of the reception of the acknowledgment frame.
Collision Detection	Collision detection from physical layer. Normal Mode or Special Purpose Mode as defined in the standard.

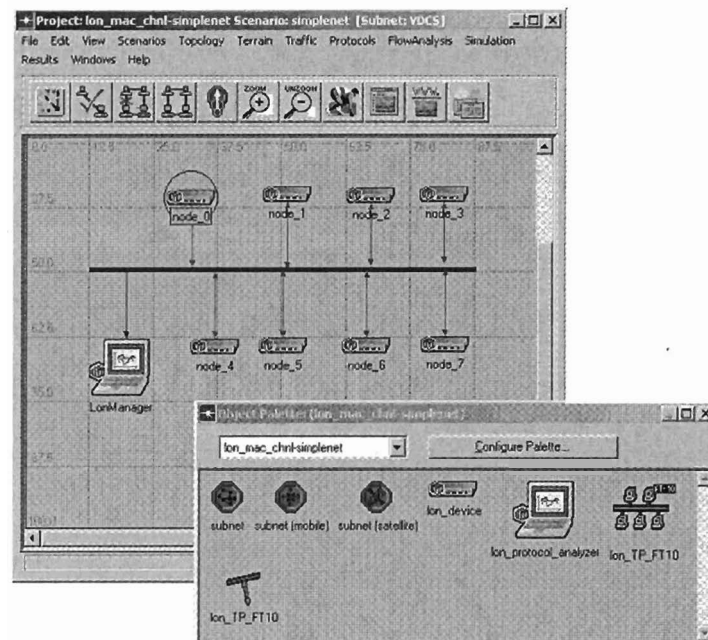


Figure A.1: LonWorks Network Analysis Toolkit OPNET module

MAC layer and higher layers being full. In other words, the rest of the channel is ideal.

A.1.3 Audience

System designers that use the LonWorks OPNET modeling tool.

A.1.4 Content

This manual provides detailed specifications of the LonWorks OPNET model, as well as an example of using this model. The example contains step-by-step procedures to teach the user how to use the model to design a LonWorks network.

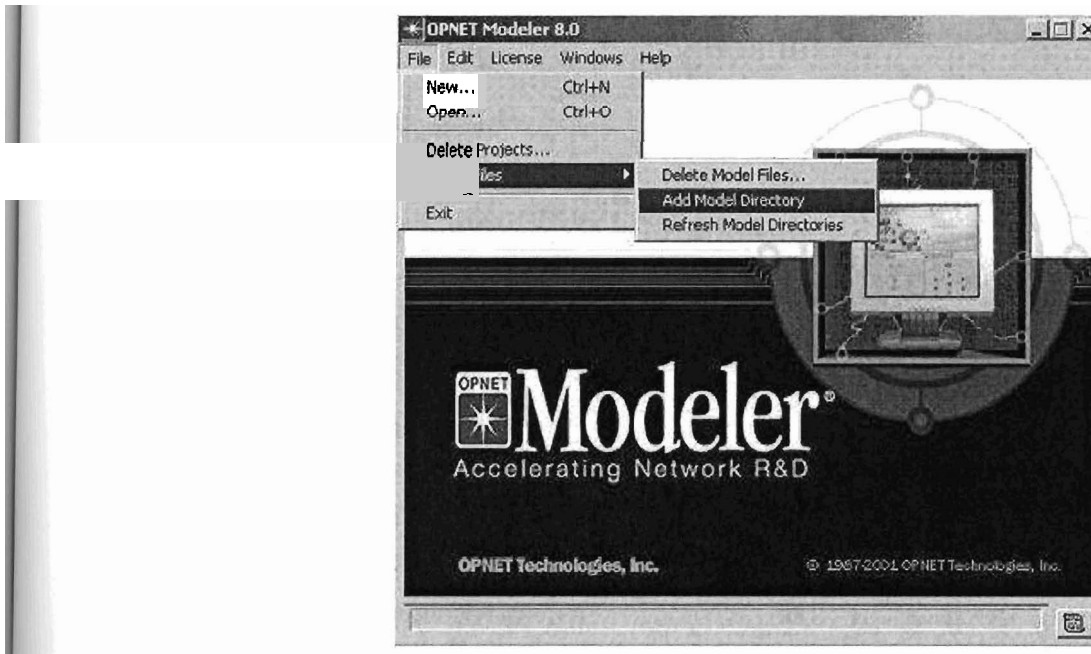


Figure A.2: Adding the LonWorks Model to Modeler

A.1.5 Related Documentation

The following publications are suggested for additional information:

1. OPNET Modeler, OPNET Modeler manuals, MIL 3, Inc. Online: <http://www.opnet.com>
2. ANSI/EIA-709.1-A, Control Network Protocol Specification, 1999
3. Fairmount Automation Inc., Development of a Virtual Distributed Control System (VDCS) Test Platform (Phase II), Final Report, Phoenixville, PA, June 2004
4. LonBuilder User Guide (078-0001-01) (see <http://www.echelon.com/support/documentation/>)

A.2 Getting Started

A.2.1 System Requirement

Hardware Requirements

- Windows NT, 2000, XP, or UNIX platform
- Minimum 800 MHz CPU
- Minimum 256 MB RAM
- Minimum of 64MB free memory

Software Requirements

- OPNET Modeler 8.0.C (Build 1283) or a more advanced version
- Microsoft Visual Studio 6.0 or a more advanced version (Windows OS only)

A.2.2 Installation

The files required in order to run the simulations are included in a single zipped file. This file must be unzipped in the user's OPNET models directory (*e.g.* “**op_models**”). After unzipping the file, choose **Model files** from the **File** menu of the *Modeler*, then click on **Add Model Directory** and select the directory of the LonWorks OPNET Model (see Figure A.12). The user should also modify the OPNET environment variable **mod_dir** (choose Preferences from the **Edit** menu), or edit manually the file “\op_admin\env_db8.0” so that the path of LonWorks OPNET model is added to the *Modeler*.

The attribute **check_newer_process_model_files** in the OPNET environment must be set to “**FALSE**” to avoid recompiling of the source code of models, which would result in simulation failure.

A.2.3 Contact information

Please direct all questions to Moshe Kam at Drexel University, m.kam@ieee.org. Mail address: M.Kam, ECE Department, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104. Phone number: (215) 895 6920.

A.3 General Overview

The LonWorks OPNET model Revision 6.1 includes scenarios (**lon_mac_VDCS.prj**) that can be used as templates. It is recommended to use these templates as the basis of user-created scenarios.

The palette in Figure A.3 can be used to add component to a new scenario. The description of these components follows:

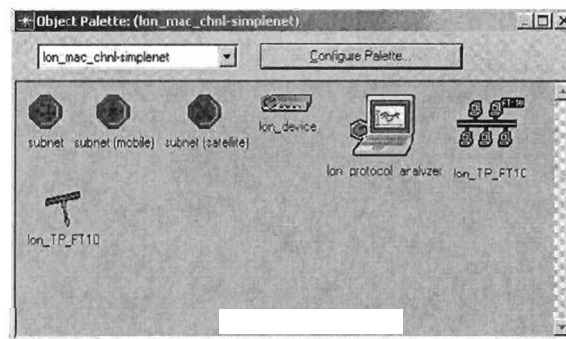


Figure A.3: LonWorks Object Palette

A.3.1 Components of the Model

A.3.1.1 lon_device model

The *lon_device* is the key component of the LonWorks OPNET model. As illustrated in Figure A.4, the LonWorks device module is comprised of the *lon_mac* process, transmitter, receiver, *lon_mac_inf* process, source, sink, and the channel streams. Module "lon_mac" implements

the p -persistent CSMA algorithm of the MAC layer. Module "lon_mac_interface" is the interface module between the MAC layer and higher layers, working as a data link layer and as part of the network layer. The solid lines between different modules represent data streams, which transmit packets or frames. The dash lines signify the status of transceivers (*busy* or *collision*).

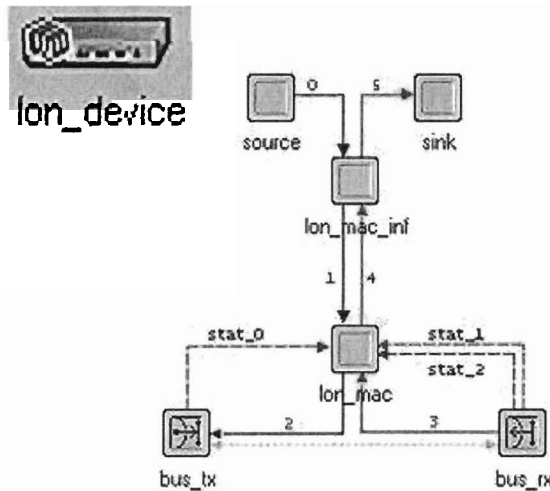


Figure A.4: lon_device_module

A.3.1.2 lon_protocol_analyzer model

The LonManager Protocol analyzer (LPA) model (Figure A.5) allows users to observe, analyze and diagnose the channel behavior of the simulated LonWorks networks. It provides the statistics of network, such as channel collision rate and error packet rate. Some of these cannot be provided by other LonWorks devices.

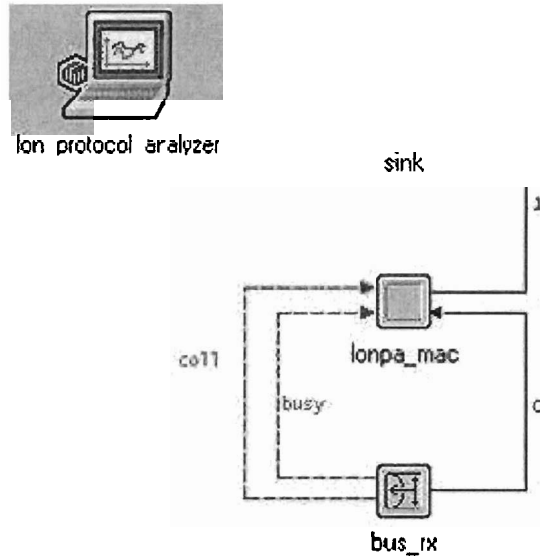


Figure A.5: lon_protocol_analyzer model

A.3.1.3 lon_TP_FT10 channel

The TP/FT-10 (Figure A.6) is the channel type connecting the LonWorks devices with arbitrary topologies. It supports network bit rates up to 78 kbps per second and 2200 meters maximum distance (bus topology).

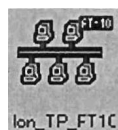


Figure A.6: TP/FT-10 Channel Model

A.3.3 Packet Format

Figure A.9 shows the frame format of the MPDU, implemented in LonWorks OPNET Model. This implementation is a slightly different from what the 709 standard specifies. Some fields such as *length*, *domain* and *full version of address format* are neglected in this format under the assumption that we are using only one subnet, one group, and one domain in the modeling tool. We only implemented the fields that affect the traffic performance and characteristics. These fields include all the MPDU headers, which are priority bit (Pri), alternative path (Alt_Path), and increment of backlog (Delta_BL). The frame format of the MPDU also includes part of the NPDU header (source and destination address and TPDU_Type). Pri, Alt_Path, and Delta_BL have the following semantics:

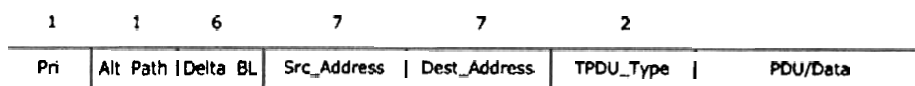


Figure A.9: Frame format of MAC layer Protocol data unit

Figure A.10 shows the packet structure implemented in the OPNET packet format editor.

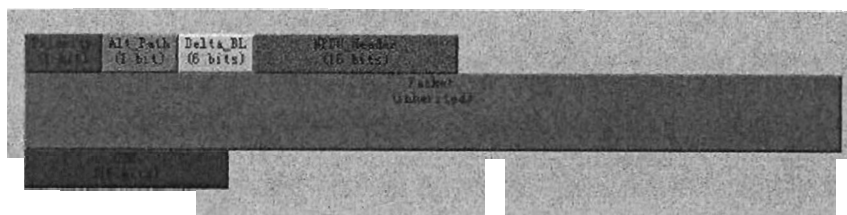


Figure A.10: Frame format from OPNET

A.3.4 Collected Statistics

The usefulness of a simulation model is dependent on the statistics of the physical system that it provides. In our LonWorks OPNET model, we have implemented a comprehensive statistical set of “points of measurements”. These statistics are stored in two types of output files: *vector* and *scalar*. *Vector* output files trace the course of a statistic over an interval of time; each data point has an associated time at which it was logged. *Scalar* output files store a set of singular statistic values grouped by simulation. Table A.2 lists the statistics being collected in vector files, while Table A.3 lists the statistics collected in scalar files. Figure A.11 shows the OPNET implementation of available statistics.

A.3.5 Features supported with this release (notation follows standard ANSI/EIA-709.1-A)

A.4 Example

An example titled `lon_mac_VDCS` has been included in the zipped file. The example includes three scenarios, namely three LonWorks simulations under different assumptions.

In this session, we present detailed step-by-step procedures, which illustrate how to build a LonWorks simulation project by using the OPNET Modeler. The goal is to demonstrate the LonWorks OPNET model.

Suppose we want to observe how the performance of the LonWorks protocols varies as a function of channel traffic. In order to do this, we design a shared channel (bus topology) that has eight nodes (8) connected on. Each node transfers standard integer-type data to its neighboring node at the various data rates (specifically, let us assume that the data rates of the nodes are 1pkts/s, 10pkts/s, 20pkts/s, 50pkts/s, 100pkts/s, 125pkts/s, 200pkts/s, 250pkts/s, and 500pkts/s, respectively). All the nodes use the UNACK service to communicate with each other.

In this example we demonstrate:

Table A.2: Statistics stored in vector output file

Name	Summary
Backlog Size	Index of backlog window size that the current node can randomly choose from before transmission while contending for the medium.
Collision Count	Number of collisions encountered by the MAC layer of this node.
End-to-End Delay (sec)	End-to-End delay of the packets accepted by this node.
Packets Offered per Node (packets)	Total number of packets sent by current node.
Packets Offered per Node (packets/sec)	Average number of packets sent by current node.
Packets Received per Node (packets)	Total number of packets received by current node.
Queue Size of Packets being held	Number of packets received from higher level being held at queue of MAC layer.
Traffic Offered per Node (bits)	Total data traffic (in bits) sent to the MAC layer from a higher layer.
Traffic Offered per Node (bits/sec)	Average data traffic (in bits) sent to the MAC layer from a higher layer.
Traffic Received per Node (bits)	Total number of bits forwarded to higher layer by the MAC layer.
Traffic Received per Node (bits/sec)	Average bits per second forwarded to the next-higher layer by the MAC layer in this node.
Transmission Attempts	Number of transmission attempts made by the MAC layer of current node before frames are successfully transmitted.

Table A.3: Statistics stored in scalar output file for LonWorks node device

Name	Summary
Average Offered Load (packets/s)	Average packet rate offered by the source as channel load.
Collision Rate (%)	Collision rate of this specific node, percentage of collisions with respect to the total number of transmission attempts.
Node Throughput (packets/s)	Throughput of this specific node.
Channel Throughput (packets/s)	Total throughput measured on the channel.
Dropped Packets (%)	Percentage of packets that are dropped due to fullness of buffer.

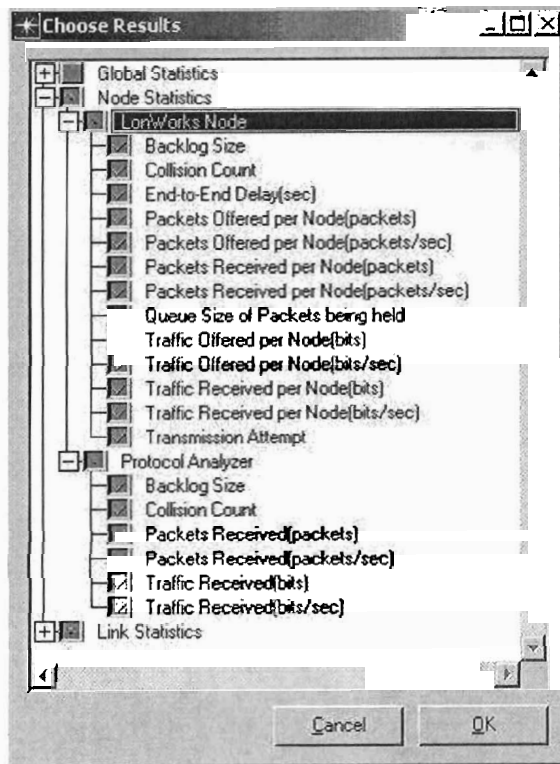


Figure A.11: Available statistics of LonWorks OPNET model

- How to design a LonWorks network simulation by using OPNET modeler.
- How to execute parametric simulations.
- How to analyze the simulated results.

A.4.1 Creating the Network Model

1. From the File menu, choose **New...**Select **Project** from the list of options, then click **OK**.
2. Name the new project **user_defined_project**, and the scenario **unack_service**.

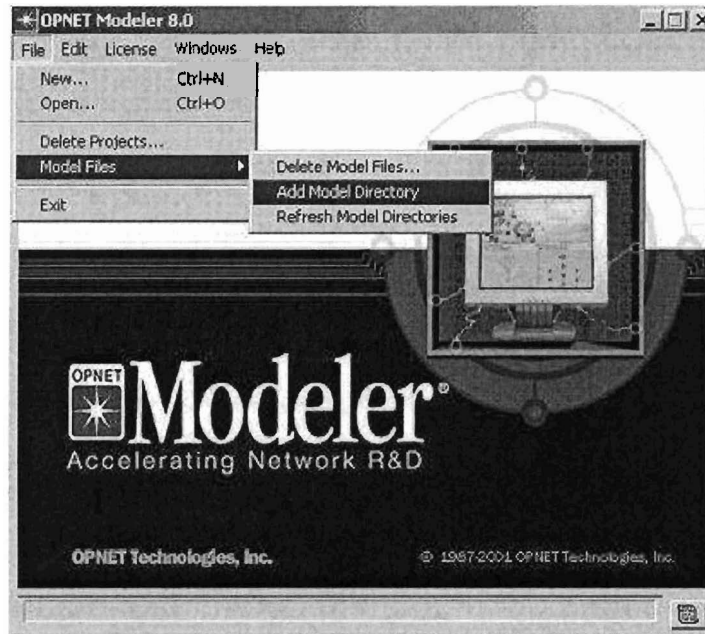


Figure A.12: Adding the LonWorks Model to Modeler

3. In the Startup Wizard, use the settings described in Table A.5.

In order to easily build the network, a custom palette is needed that contains the necessary objects for the network. To create the palette:

1. In the **Object Palette** window, click on the **Configure Palette...** button.
2. In the **Configure Palette** dialog box, click **Clear** (All objects except the subnet are

Table A.4: Initial settings of sample scenario

Dialog Box Name	Value
Initial Topology	Default Values: Create Empty Scenario
Choose Network Scale	Office("Use Metric Units" enabled)
Specify Size	100m x 100m
Select Technologies	None
Review	Check values, then click ok

removed from the palette.)

3. Click on the Node Models button; then add **lon_device** and **lon_procotol_analyzer** from the list of available node models. Click **OK** to close the dialog box when finished.
4. Click on the Link Models button; then add **lon_TP_FT10** from the list of available link models. Click **OK** to close the dialog box when finished.
5. Save the **Object Palette** by clicking on the Save button in the **Configure Palette** dialog box. Use **lon_example_palette** as the file name.
6. Click **OK** to close the **Configure Palette** dialog box (the **lon_example_palette** Object Palette is ready for use).

Instead of creating the entire network by hand, one can use rapid configuration, as follows:

1. Choose **Rapid Configuration** from the **Topology** menu.
2. Select **Bus** from the menu of available configurations, then click **OK**.
3. In the **Rapid Configuration: Bus** dialog box, set the following values as shown in Figure A.13.
4. Click **OK** when all the values are entered. (The network on the following is drawn in the workspace.)

In order to analyze the channel behavior of the task, one needs to add a **lon_protocol_analyzer** module into the network:

1. Click and drag the protocol analyzer from the palette into the left side of the tool area.

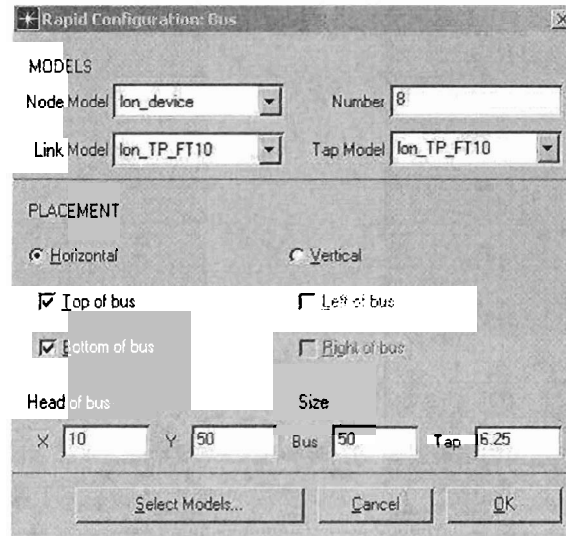


Figure A.13: Rapid Configuration

2. Click on the lon_TP_FT10 tap link in the palette, and then draw a tap from the bus to the protocol analyzer node. (Note: you cannot draw a tap from the node to the bus.)

The completed bus model looks like the diagram shown in Figure A.15

After completing the network construction for the example, one must specify the node attributes by select **Edit Attributes** from the pop-up menu after right-clicking the mouse button when the mouse is pointed to any one of the selected LonWorks nodes (node_0 to node_7). To apply the changes to all of the nodes:

1. Check the **Apply Changes to Selected Objects** check box in the node attributes dialog box.
2. Click **OK** to close the Attributes dialog box.

In order to display the network statistics that the nodes record during the simulation, we need to enable one (and only one) of the recording interrupts in the node attributes. To

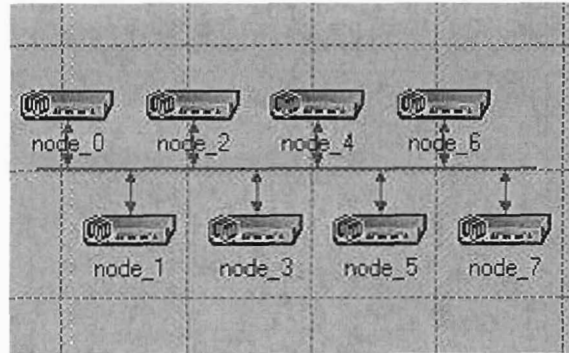


Figure A.14: Network Created by Rapid Configuration

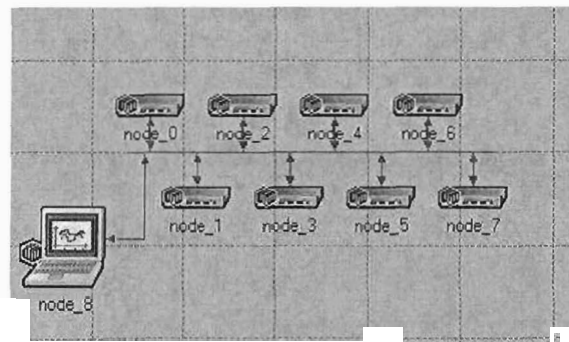


Figure A.15: Adding Protocol Analyzer

do this, one selects any one of the nodes; and changes the attribute `lon_mac.endsim intrpt` in the window of node attributes to **enabled**.

Finally, one can save the model (but not exit the Project Editor) and close the object palette.

A.4.2 Collecting Statistics

To collect statistics:

1. Right-click in the workspace to display the workspace pop-up menu, and select

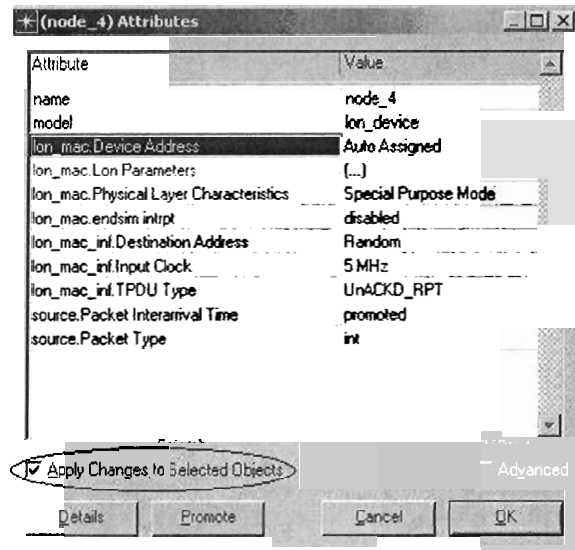


Figure A.16: Node Attributes Dialog Box

Choose Individual Statistics.

- Place checks in the check boxes. In this example, we assume we would like to see the statistics of **Backlog Size**, **Packets Offered per Node**, **Packets received per Node**, **Queue Size of Packets being held** and some other related statistics.
- Click **OK** to close the Choose Results dialog box.

A.4.3 Executing the Simulation

Our simulation produces both scalar and vector results. An output scalar file and an output vector file must be specified where these results accumulate from successive simulations. This operation is done in the Simulation Configuration Editor, which can be reached by selecting **Configure Simulation (advanced)** from the Simulation menu in the Project Editor.

The goal of this session is to observe how the performance of the protocols varies as a function of channel traffic. The inter-arrival time input parameter will be varied in a

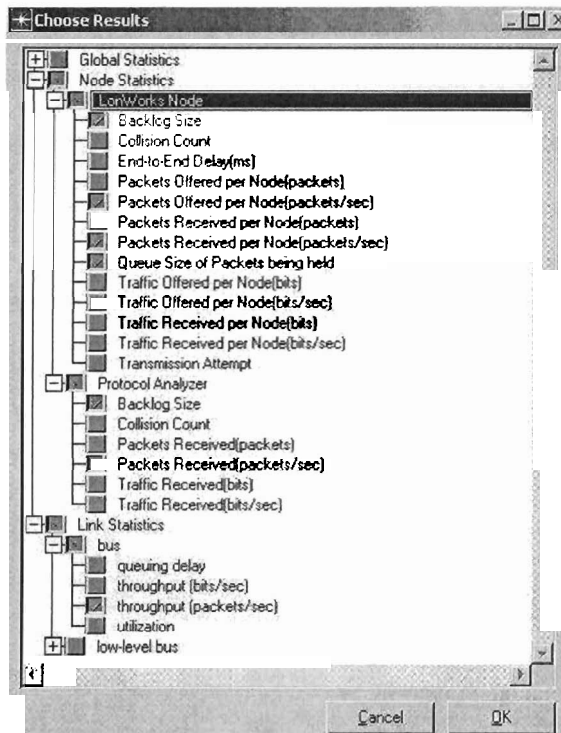


Figure A.17: Choosing Available Statistics

series of simulations to produce different levels of traffic and hence, different levels of throughput. Conclusions will be drawn from the results of nine simulations, each with a different inter-arrival time value.

1. Choose **Configure Simulation (advanced)** from the Simulation menu in the Project Editor.
2. Right-click on the simulation set icon and select **Edit Attributes** from the Object pup-up menu.
3. Set the Scalar file to **lon_example_scalar_file**

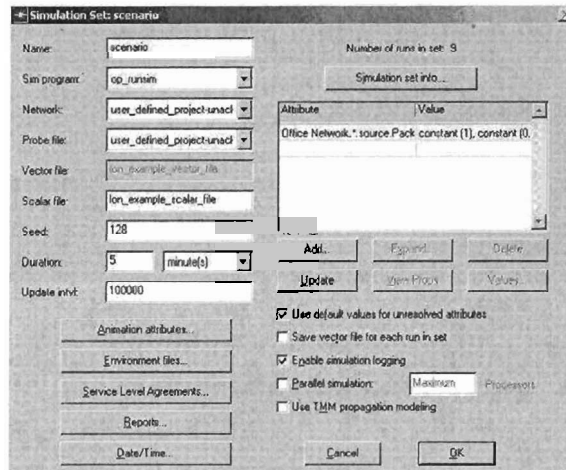


Figure A.19: Simulation Set

Table A.5: Initial settings of sample scenario

<p>Simulation Completed - Collating Results. Events: Total(10650851), Average Speed (19048 events/sec) Time: Elapsed (9min. 19 sec.), Simulated (5 min. 0 sec) Simulation Log: 1 entries</p>

3. The ten simulations display their progress as they execute. Any simulation run will be no longer than 5 minutes (simulation time) and will terminate with a message such as that shown in

4. When the simulations are complete, close the editor.

A.4.4 Analyzing the simulation results

Once the simulation finished executing, one may want to examine the collected network statistics. These statistics are stored in two formats.

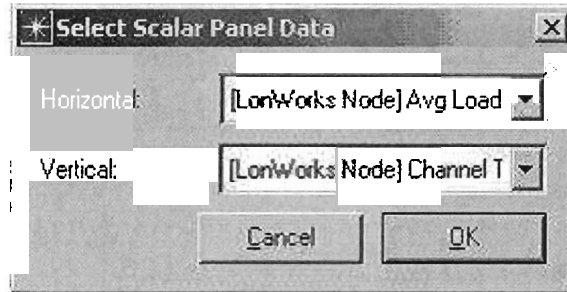


Figure A.20: Select Scalar Panel Data

A.4.4.1 Analyzing the results stored in scalar file

To open the scalar output file:

1. In the Project Editor, choose **View results (Advanced)** from the **Results** menu
2. Select **Load Output Scalar file** from the **File Menu**.
3. Select **lon_example_scalar_file** from the list of available files.
1. Click on the **Create a graph of two scalars** action button.
2. Select the horizontal variable **[LonWorks Node] Avg load offered by Each Node (packets/sec)** first, and then select the vertical variable **[LonWorks Node] Channel Throughput (packets/sec)** from the menu of available scalars that pops up.
3. Click **OK**

The graph of the scalar panel appears in the workspace as shown in Figure A.21.

Instead of using **[LonWorks Node] Channel Throughput (packets/sec)** as the vertical variable, you can choose **[Protocol Analyzer] Collision Rate of Channel (%)** as the vertical variable, which shows the collision probability of channel versus packet inter-arrival times. It appears as in Figure A.22:

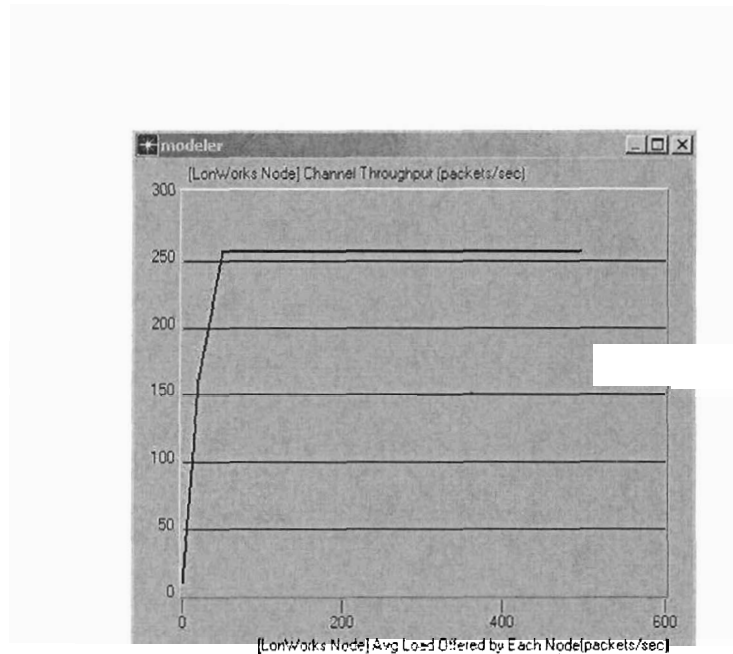


Figure A.21: Simulation Result 1

One can get other plots of available statistics by repeating steps 2 and 3 using other horizontal/ vertical variables.

A.4.4.2 Analyzing the results stored in vector file

To view the statistics results for the simulation gathered by each node:

1. Right-click on the LonWorks device node choose View Results from the pop-up menu
2. Expand the Office Network:node_0:LonWorks Node hierarchy.
3. Click on the boxes next to **Backlog Size**, **Queue Size of Packets being held**, **Packets Received per Node (packets/sec)** and **Packet Offered per Node (packets/sec)** to indicate that you want to view those results.
4. Click the Show button in the View Results dialog box.

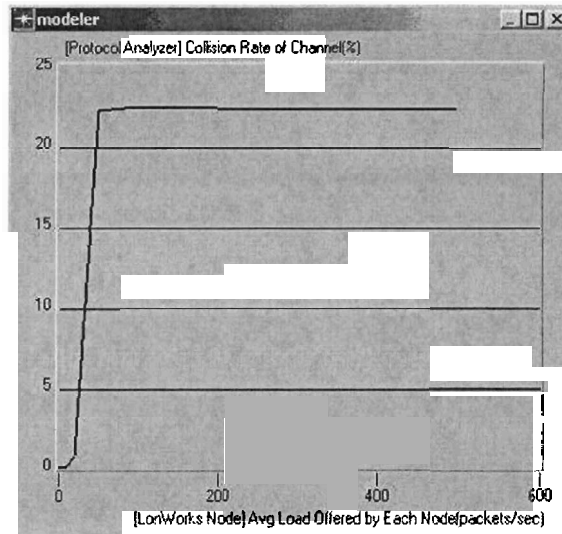


Figure A.22: Simulation Results 2

The graph of the network statistics gathered by specified node appears in the Project Editor:

In order to view the same statistic for all the nodes by once, follow the steps:

1. Left-click on the create a graph of a statistic action button (The view Results dialog box opens).
2. Expand the following hierarchy **user_defined_project-unack_service:Object Statistics:Office Network**.
3. Select the statistic you want to study from each node (for example, select **Backlog Size**).
4. Click on the **Show** button

The graph of the backlog size for each LonWorks node versus time progresses appears in the Project Editor:

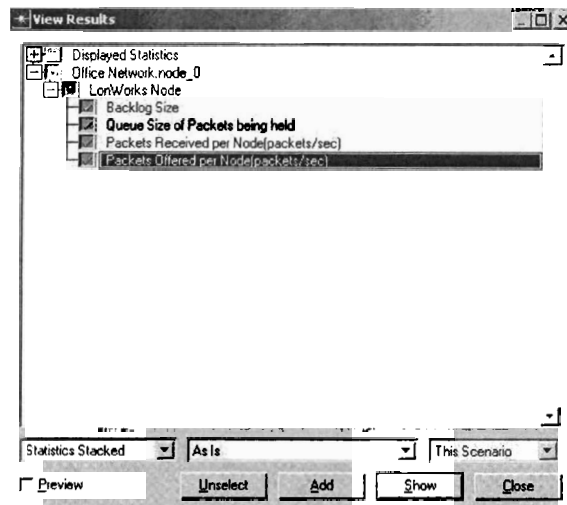


Figure A.23: Displayed Statistics

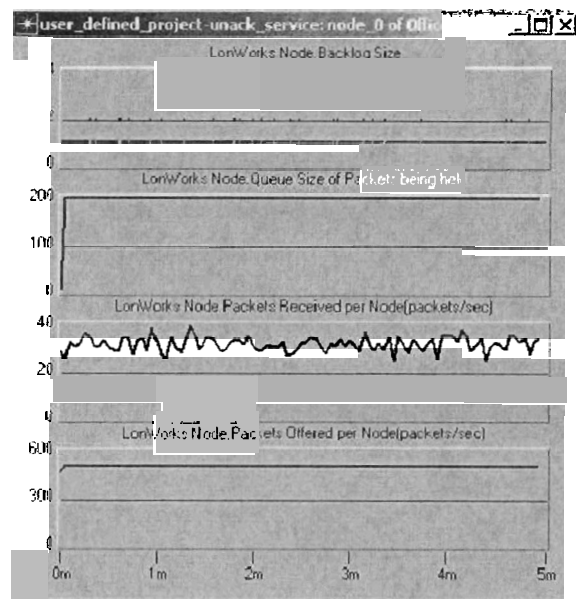


Figure A.24: Simulation Results 3

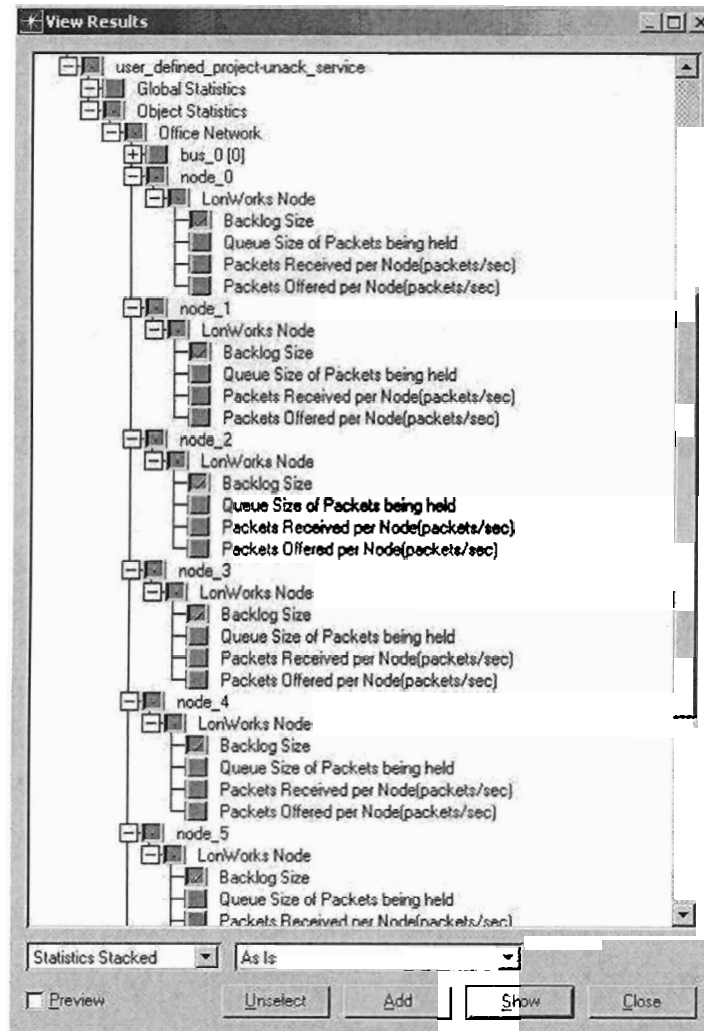


Figure A.25: Displayed Statistics

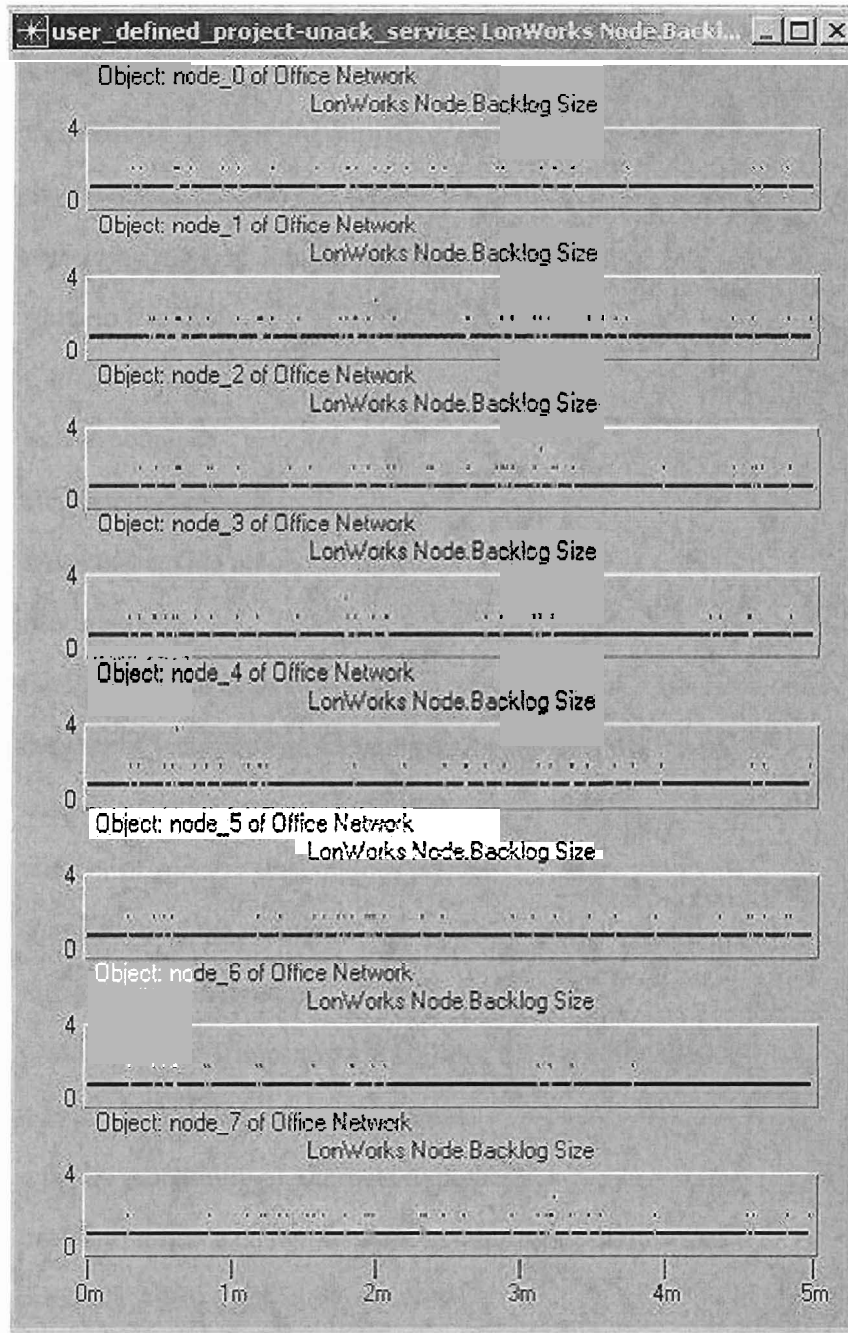


Figure A.26: Simulation Results 4

Nomenclature

a_k :	a steering factor
$b_{i,j}$:	the notation representing $\lim_{t \rightarrow \infty} P\{s(t) = i, b(t) = j\}$
$b(t)$:	the stochastic process that represents number of remaining backlog waiting period before the sensor starts to transmit
C :	<i>Collision</i> state
H_0 :	null hypothesis
H_1 :	alternate hypothesis
i :	index of backlog stage
I :	<i>Idle</i> state
N :	number of local sensors connected on the channel
n :	number of local sensors transmitting on the channel
n_I :	number of <i>idle</i> slots on the channel
n_S :	number of <i>success</i> slots on the channel
n_T :	summation of the number of the <i>collision</i> and <i>idle</i> slots
m :	maximum backlog value
p :	probability that a transmission collides for a specific sensor
P_0 :	<i>a priori</i> probability of null hypothesis
P_1 :	<i>a priori</i> probability of alternate hypothesis
$P_c^{(w)}$:	probability that a collision happened during w -th time slot
$P_{ch_{coll}}$:	probability that a collision occurred

$P\{i, j m, n\}$:	the notation representing $P\{s(t+1) = i, b(t+1) = j s(t) = m, b(t) = n\}$
$P_i^{(w)}$:	probability that no sensor would attempt to get access during w -th time slot
P_f :	false alarm probability
$P_{n_S, n_I, n_C n}$:	probability density function of n_S , n_I , and n_C , given there are n sensors attempt to transmit
P_m :	missed detection probability
P_s :	probability that an occurring transmission is successful
$P_s^{(w)}$:	probability that a single sensor would gain access to the channel during w -th time slot
P_{tr} :	the probability that at least one sensor attempts to transmit in any given time slot on the channel
$P_{C n}$:	the probability that a randomly selected time slot will be a <i>collision</i>
P_D :	global probability of detection
P_F :	global probability of false alarm
$P_{I n}$:	the probability that a randomly selected time slot will be a <i>idle</i>
$P_{S n}$:	the probability that a randomly selected time slot will be a <i>success</i>
$s(t)$:	the stochastic process representing the backlog stage $(0, \dots, m)$ for a specific local sensor
S :	<i>Success</i> state
u_0 :	decision made by the Data Fusion Center
u_i :	decision made by the i -th local sensor
u_i^T :	decision made by the i -th local sensor during time T

- W_{base} : a design parameter of base window size
 W_i : current contention window size
 x_l : data vector of of the complete data set χ
 x_l^m : data vector of of the missing data set χ^m
 x_l^o : data vector of of the incomplete data set χ^o
 $y_k(t)$: observation made by the k -th sensor
 z_i^T : observation made by the i -th local sensor during T time stamp
 τ : probability that a sensor transmit in a randomly chosen time
 χ : complete data set
 χ^m : missing data set
 χ^o : observed data set (the incomplete data)
 κ : a vector of unknown parameters
 ω_0 : center frequency of the signals
 $\eta(t)$: white Gaussian noise with zero mean
 Ψ : a random vector corresponding to the observed data ψ
 Υ^i : the "incomplete" sample space
 Υ^c : the "complete" sample space

Bibliography

- [1] *IEEE 802.3 CSMA/CD (Ethernet) Standard*. Available On-line: <http://grouper.ieee.org/groups/802/11/>.
- [2] *Introduction to the LonWorks System*. Available on-line: <http://osa.echelon.com/Program/PDFs/IntroLonWorksSystem.pdf>.
- [3] *OPNET Modeler*. Mill 3, Inc.
- [4] *SmartControls, LLC*. Available On-line: <http://www.smartcontrols.com/home.htm>.
- [5] *Wireless Local Area Network Standard*. Available On-line: <http://www.ieee802.org/3>.
- [6] *ANSI/EIA-709.1-A, Control Network Protocol Specification*. Echelon Inc., San Jose, USA, 1995.
- [7] M. D. Adams, H. Hu, and P. J. Probert. Towards a real-time architecture for obstacle avoidance and path planning in mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 584–589, March 1990.
- [8] B. C. Arrue, A. Ollero, and J. R. M. de Dios. An intelligent system for false alarm reduction in infrared forest-fire detection. *IEEE Intelligent Systems*, 15:64–73, May/June 2000.
- [9] Z.D. Bai, P.R. Krishnaih, and L.C. Zhao. On rates of convergence of efficient detection criteria in signal processing with white noise. *IEEE Trans. Info. Theory*, 35(2):380–388, 1989.
- [10] A.J. Barbell. Improving the resolution performance of eigenstructure -based direction-finding algorithm. In *In Proceeding of The International Conference on Acoustics, Speech, and Signal Processing*, pages 336–339, 1983.
- [11] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communication*, 18(3), March 2000.
- [12] M. Brandstein and D. Ward. *Microphone arrays*. Springer-Verlag, 2001.
- [13] R. Brooks and S. Iyengar. **Maximizing multi-sensor system dependability**. In *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 1–8, Killarney, Ireland, 1996.
- [14] Z. Chaczko and F. Ahmad. Wireless sensor network based system for fire endangered areas. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on*, volume 2, pages 203–207, July 2005.

- [15] Z. Chair and P.K. Varshney. Optimal data fusion in multiple sensor detection systems. *IEEE Transactions on Aerospace and Electronic Systems*, 22(1):98–101, 1986.
- [16] X. Chen and G. Hong. A simulation study of the predictive p -persistent csma protocol. In *Proceedings of 35th Annual Simulation Symposium*, pages 345–351, 2002.
- [17] S. Choi, B. Kim, J. Park, C. Kang, and D. Eom. An implementation of wireless sensor network. *Consumer Electronics, IEEE Transactions on*, 50(1):236–244, 2004.
- [18] C. Chong and S.P. Kumar. Sensor networks: evolution, opportunities, and challenges. In *Proceedings of the IEEE*, volume 91, pages 1247–1256, August 2003.
- [19] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J.R. Stat. Soc. B*, 39(1), 1977.
- [20] J. Deng, R. Han, and S. Mishra. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *Dependable Systems and Networks, 2004 International Conference on*, volume 1, pages 637–646, June 2004.
- [21] S. Haykin, J.P. Reilly, V. Kezys, and E. Vertatschitsch. Some aspects of array signal processing. *Spatial Statistics and Digital Image Analysis*, 1992.
- [22] M. Hefnawi and G.Y. Delisle. Smart antenna system for wideband cdma signals. In *Antennas and Propagation Society International Symposium, 2001. IEEE*, volume 4, pages 22–25, July 2001.
- [23] R. Hill, R. Bates, and S. Waters. On centrohermitian matrices. *Siam J. Matrix Anal. Appl.*, 11:128–133, Jan 1990.
- [24] A. Houles and Y. Bar-Shalom. Multisensor tracking of a maneuvering target in clutter. *IEEE Transactions on Aerospace and Electronic Systems*, 25(2):176–189, 1989.
- [25] S. Jeng, C. Huang, and T. Huang. Performance evaluation of doa based beamforming in w-cdma system. In *IEEE Vehicular Technology Conference*, volume 7-11, pages 2652–2656, 1983.
- [26] M. Kam, W. Chang, and Q. Zhu. Hardware complexity of binary distributed detection systems with isolated local bayesian detectors. *IEEE Trans. Systems, Man, and Cybernetics*, 21(3):565–571, May/June 1991.
- [27] M. Kam and E. Lin. Writer identification using hand-printed and non-hand-printed questioned documents. *Journal of Forensic Science*, 48(6):1391–1395, November 2003.
- [28] L. Kleinrock and F.A. Tobagi. Packet switching in radio channel. i. carrier sense multiple-access models and their throughput-delay characteristics. *IEEE, Trans, on Communication*, COM-23(12), December 1975.

- [29] R. Kumaresan and D.W. Tufts. Estimating the angles of arrival of multiple plane waves. *IEEE Trans. Aero. and Elec.*, 19(1):134–138, 1983.
- [30] F. Li. A method for detection of deformations in large phased array antennas for spaceborne synthetic aperture radars. *IEEE Transactions on Antennas and Propagation*, 32(5):512–517, 1984.
- [31] Hsueh-Jyh Li and Ta-Yung Liu. Comparison of beamforming techniques for w-cdma communication systems. *Vehicular Technology, IEEE Transactions on*, 52(4):752–760, July 2003.
- [32] E. Lin, L. Bai, and M. Kam. Efficient doa estimation method employing unitary improved polynomial rooting. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 257–260, May 2004.
- [33] E. Lin, M. Wang, and M. Kam. Modeling and performance evaluation of collision resolution algorithm for lonworks control networks. In *Proceedings of IEEE International Conference on Networking, Sensing and Control*, March 2005.
- [34] E. Lin, E. Woertz, and M. Kam. Lsb steganalysis using support vector regression. In *Proceedings of the IEEE information Assurance Workshop*, June 2004.
- [35] D.A. Linebarger, R.D. Degroat, and E.M. Dowling. Efficient direction-finding methods employing forward-backward averaging. *IEEE Trans. Signal Processing*, 42:2136–2145, 1994.
- [36] M. Longo, T.D. Lookabaugh, and R.M. Gray. Quantization for decentralized hypothesis testing under communication constraints. *IEEE Trans. Inf. Theory*, 36(2):241–255, March 1990.
- [37] K. Lorincz, D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor networks for emergency response: challenges and opportunities. *Pervasive Computing, IEEE*, 3(4):16–23, Oct-Dec 2004.
- [38] J.C. Lupo. Defense applications of neural networks. *Communications Magazine, IEEE*, 27(11):82–88, 1989.
- [39] Y. Mao, F.R. Kschischang, B. Li, and S. Pasupathy. A factor graph approach to link loss monitoring in wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 23(4):820–829, 2005.
- [40] R. Min and etc. Lowpower wireless sensor networks. In *VLSI Design 2001*, January 2001.
- [41] M. Miskowicz and et al. Performance analysis of predictive p -persistent csma protocol for control networks. In *4th IEEE International Workshop on Factory Communication Systems*, pages 249–256, Vasteras, Sweeden, August 2002.

- [42] N. Mladineo and S. Knezic. Optimisation of forest fire sensor network using gis technology. In *Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on*, pages 391–396, June 2000.
- [43] R. Nowak, U. Mitra, and R. Willett. Estimating inhomogeneous fields using wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 22(6):999–1006, August 2004.
- [44] M. Pesavento, A. Gershman, and M. Haardt. Unitary root-music with a real-valued eigendecomposition: A theoretical and experimental performance study. *IEEE Trans. Signal Processing*, 48(5):1306–1314, May 2000.
- [45] S. Pillai and B. Kwon. Forward-backward spatial smoothing techniques for the coherent signal identification. *IEEE Trans. Acoust., Speech, Signal Processing*, 37:8–15, Jan 1989.
- [46] G. Pottie. Wireless sensor networks. In *Information Theory Workshop*, volume 7-11, page 139C40, Killarney, Ireland, June 1988.
- [47] G. Pottie and W. Kaiser. Wireless integrated network sensors. *Comm. ACM*, 43, May 2000.
- [48] C. Rago, P. Willett, and Y. Bar-Shalom. Censoring sensors: a low-communication-rate scheme for distributed detection. *IEEE Transactions on Aerospace Electronic and Systems*, 32(2):554–568, April 1996.
- [49] A. R. Reibman and L.W. Nolte. Design and performance comparison of distributed sensor system. *IEEE Transactions on Aerospace and Electronic Systems*, 23(6):789–797, 1987.
- [50] A. R. Reibman and L.W. Nolte. Optimal detection and performance of distributed sensor systems. *IEEE Transactions on Aerospace and Electronic Systems*, 23(1):24–30, 1987.
- [51] R. Roy, A. Paulraj, and T. Kailath. Esprit-estimation of signal parameters via rotational invariance techniques. *IEEE Trans. Acoust., Speech, and Signal Processing*, 37(7):984–995, 1989.
- [52] R.O. Schmidt. Multiple emitter location and signal parameter estimation. In *Proc. RADC Spectral Estimation Workshop*, pages 243–258, 1979.
- [53] M. Sveda and R. Vrba. Integrated smart sensor networking framework for sensor-based appliances. *Sensors Journal, IEEE*, 3(5):579–586, 2003.
- [54] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *Communications, IEEE Transactions on*, 32(3):246–257, 1984.

- [55] R.R. Tenney and N.R. Sandell. Detection with distributed sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 17(4):501–509, 1981.
- [56] S. C. Thomopoulos, R. Viswanathan, and D.K. Bougoulias. Optimal decision fusion in multiple sensor systems. *IEEE Transactions on Aerospace and Electronic Systems*, 23(5):644–653, 1987.
- [57] S. C. Thomopoulos, R. Viswanathan, and D.K. Bougoulias. Optimal distributed decision fusion. *IEEE Transactions on Aerospace and Electronic Systems*, 25(5):761–765, 1989.
- [58] A. Tiwari, F.L. Lewis, and S.S. Ge. Wireless sensor network for machine condition based maintenance. In *Control, Automation, Robotics and Vision Conference*, volume 1, pages 461–467, Dec 2004.
- [59] J.N. Tsitsiklis. Decentralized detection. in *Advances in Statistical Signal Processing*, 1990.
- [60] P. K. Varshney. *Distributed Detection and Data Fusion*. Springer-Verlag, New York, 1997.
- [61] R. Viswanathan and P. K. Varshney. Distributed detection with multiple sensors: part i - fundamentals. *Proceedings of the IEEE*, 85(1):45–63, January 1997.
- [62] E. Waltz and J. Llinas. *Multisensor Data Fusion*, chapter 6. Artech House, Boston, MA, 1990.
- [63] C. M. Wang. Location estimation and uncertainty analysis for mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 1230–1235, 1988.
- [64] M. Wang, E. Lin, E. Woertz, and M. Kam. Collision resolution simulation for distributed control architectures using lonworks. In *Proceedings of IEEE International Conference on Automation Science and Engineering*, July 2005.
- [65] T. Wang, P.K. Han, Y.S.; Varshney, and P. Chen. Distributed fault-tolerant classification in wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 23(4):724–734, 2005.
- [66] J.F. Woolett. MMWR/FLIR/ATR sensor fusion: proof of concept. *Aerospace and Electronic Systems Magazine, IEEE*, 3(6):22–25, 1988.
- [67] Y Yuan. *Decision Fusion in Distributed Detection and Bioinformatics*. PhD thesis, Drexel University, Philadelphia, PA, 2004.
- [68] Y. Yuan and M. Kam. Distributed decision fusion for sensor network applications. *IEEE Tran. Instrumentation and Measurement Technology*, 53(4):1339–1344, August 2004.

- [69] Q. Zhu, X. Zhu, and M. Kam. A simple algorithm in adaptive decision fusion. In *Proceedings of the 1995 American Control Conference*, pages 804–805, Baltimore, MD, June 1994.
- [70] X. Zhu. *Decision-level Data Fusion*. PhD thesis, Drexel University, Philadelphia, PA, 1997.
- [71] X. Zhu and M. Kam. Divergence of the adaptive decision fusion scheme of Ansari *et al.* *IEEE Transactions on Aerospace and Electronic Systems*, under review.
- [72] Y. Zhu and X. R. Li. Unified fusion rules for multisensor multihypothesis network decision systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 33(4):502–513, 2003.

Vita

Erwei Lin was born in Guangdong, China in 1977. He received his B.S. and M.S. degrees from the Electrical and Computer Engineering Department, College of Engineering, Drexel University in 2001. His research topic is in the area of distributed sensor network, directional-of-arrival (DOA), and signal processing. Mr. Lin is the 2004-2005 recipient of the Outstanding Graduate Student Award. His main publications include:

1. Erwei Lin, Li Bai, and Moshe Kam, Efficient DOA Estimation Method Employing Unitary Improved Polynomial Rooting, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 257-60, 17-21 May, 2004.
2. Moshe Kam, and Erwei Lin, Writer Identification using Hand-printed and Non-hand-printed Questioned Documents, *Journal of Forensic Science*, Vol. 48, Issue 6, pp. 1391-1395 November, 2003.
3. Erwei Lin, Mianyu Wang, and Moshe Kam, Modeling and Performance Evaluation of Collision Resolution Algorithm for LonWorks Control Networks, *Proceedings of IEEE International Conference on Networking, Sensing and Control*, 2005, 19-22 March, 2005, Tucson, Arizona.
4. Lit-Hsin Loo, Erwei Lin, Moshe Kam, and Pramod Varshney, Cooperative Multi-agent Constellation Formation under Sensing and Communication Constraints, *Cooperative Control and Optimization*, Kluwer Academic Publishers, pp.143-169, 2001
5. Erwei Lin, Edward Woertz, and Moshe Kam, LSB Steganalysis Using Support Vector Regression, Proceedings of the IEEE information Assurance Workshop, 10-11 June 2004 at United States Military Academy, West Point, New York.
6. Mianyu Wang, Erwei Lin, Edward Woertz, Moshe Kam, Collision Resolution Simulation for Distributed Control Architectures Using LonWorks, *Proceedings of IEEE International Conference on Automation Science and Engineering*, 2005
7. Erwei Lin, Finding and Engaging Surface Targets by a Cohort of Cooperative Agents, M.S. Thesis, Drexel University, September, 2001
8. Erwei Lin, and Moshe Kam, Detection in Distributed Sensor Network Using Single Random Access Channel, *submitted to IEEE Transactions on Systems, Man and Cybernetics*, 2005.