Autoregression Models for Trust Management in Wireless Ad Hoc Networks

by

Zhi Li

Thesis submitted to the

Faculty of Graduate and Postgraduate Studies

In partial fulfillment of the requirements

For Masters of Applied Science degree in

Electrical Engineering

School of Electrical Engineering & Computer Science

Faculty of Engineering

University of Ottawa

Abstract

In this thesis, we propose a novel trust management scheme for improving routing reliability in wireless ad hoc networks. It is grounded on two classic autoregression models, namely Autoregressive (AR) model and Autoregressive with exogenous inputs (ARX) model. According to this scheme, a node periodically measures the packet forwarding ratio of its every neighbor as the trust observation about that neighbor. These measurements constitute a time series of data. The node has such a time series for each neighbor. By applying an autoregression model to these time series, it predicts the neighbors future packet forwarding ratios as their trust estimates, which in turn facilitate it to make intelligent routing decisions. With an AR model being applied, the node only uses its own observations for prediction; with an ARX model, it will also take into account recommendations from other neighbors. We evaluate the performance of the scheme when an AR, ARX or Bayesian model is used. Simulation results indicate that the ARX model is the best choice in terms of accuracy.

Acknowledgement

I would like to show my sincerest gratitude to Prof. Amiya Nayak, my supervisor, for his exceptional guidance and support not only in this thesis but also in my study and research. I am grateful to my co-supervisor Prof. Ivan Stojmenovic for his help in my study. Special thanks to Dr. Xu Li who gave me valuable advices, suggestions and encouragements.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Contents

List of Figures

List of Tables

List of Acronyms

AIC Akaike Information Criterion

AR Autoregressive

ARX Autoregressive with exogenous inputs

BIC Bayesian Information Criterion

BRSN Beta Reputation system for Sensor Networks model

CA Certificate Authority

CLA Central Authentication

DARWIN Distributed and Adaptive Reputation mechanism for Wireless ad hoc Networks

FPE Final Prediction Error

KIC Kullback Information Criterion

LS Least-Square

MANET Mobile Ad hoc Network

MDL Minimum Description Length

P2P Peer-to-Peer

PKI Public Key Infrastructure

TM Trust Management

TFN Triangular Fuzzy Number

YW Yule-Walker

Chapter 1

Introduction

1.1 Background and Motivation

With the wireless devices such as cell phones and laptops are becoming more and more popular, the desire of reliable wireless communication is increasing. Multi-hop, wireless, ad hoc networking has been the focus of many recent research and development efforts for its applications in military, commercial, and educational environments such as wireless LAN connections in the office, networks of appliances at home, and sensor networks. Wireless ad hoc networks can be flexibly and quickly deployed for many applications such as automated battlefield operations, search and rescue, and disaster relief. Unlike wired networks or cellular networks, no physical backbone infrastructure is installed in wireless ad hoc networks. Operating in ad-hoc mode allows the wireless devices within range of each other to discover and communicate in peer-to-peer fashion without involving central access points. A communication session is achieved either through a single-hop radio transmission if the communication parties are close enough, or through relaying by intermediate nodes otherwise. Based on the different applications requirements, the wireless ad hoc network could be further extended to Mobile Ad Hoc Network (MANET) with the ability to rebuild the route links all the time for nodes mobility, Wireless Mesh Network (WMN) with the mesh topology and

Wireless Sensor Network (WSN) with a large amount of sensors spatially distributed in a physical environment.

Wireless ad hoc networks are open, self-organized and autonomous [36]. Routing in such networks, for example, is based on a full trust/cooperation between all the participating hosts. Correct transport of the packets on the network relies on the veracity of the information given by the other nodes. The emission of false routing information by a host could thus create bogus entries in routing tables throughout the network making communication difficult. Furthermore, the delivery of a packet to a destination is based on a hop by hop routing and thus needs total cooperation from the intermediate nodes. A malicious host could, by refusing to cooperate, quite simply block or modify the traffic traversing it. Wireless ad hoc networks cannot avoid malicious and selfish nodes. When these nodes are present, the network is at high risk of being threatened and attacked by them. In order to continue working, the network needs to prevent these malicious and selfish nodes from breaking the network. Trust management [7] aims to help capture node relations and predict node behaviors so as to avoid the future transactions of untrustworthy nodes.

Trust is therefore a fundamental issue in wireless ad hoc networks for effective and secure communication. Originally trust is used in the social life to represent a relationship between people. Trust in wireless ad hoc or peer-to-peer networks is derived from the definition in social domains. For various applications in such networks, the trust definitions are focused in different points. According to Eschenauer et al. [15], trust is defined as "a set of relations among entities that participate in a protocol". These relations are based on the evidence generated by the previous interactions of entities within a protocol. In general, if the interactions have been faithful to the protocol, then trust will accumulate between these entities.

1.2 Objectives and Contributions

There are a number of trust management systems introduced in the literature to deal with trust issues. They can be simply divided into two main categories: Credential-Based Trust Management Systems and Reputation-Based Systems. In most Credential-Based Trust Management Systems, like PolicyMaker [9], KeyNote [8] and Public Key Infrastructure (PKI) [41], the trust relationships between entities are established by managing and exchanging credentials which should be verified and restricted by a preset policy. An entity will be trusted after its credentials have been verified. Credential-Based Trust Management Systems usually make a binary decision whether to allow the request or not based on whether its requestor is trustworthy or not. Because of this binary approach, Credential-Based Trust Management Systems lack flexibility. On the other hand, the Reputation-Based Trust Management Systems perform better by focusing on the evaluation of trust value. A reputation-based system calculates the reputation value of a node by gathering observations of node behaviors. Many schemes for reputation calculation exist, for example, EigenRep [24], Agent-based Trust and Reputation Management (ARTM) [10], TrustMe [37]. Most of the existing trust management systems are not well suitable for wireless ad hoc networks. The objective of this thesis is to explore viable alternatives for modeling trust.

In this thesis, in line with the reputation-based approach, we propose a novel trust management scheme as viable alternative for reliable routing in wireless ad hoc networks, by exploiting two classic autoregression models [11], i.e., Autoregressive (AR) model and Autoregressive with exogenous inputs (ARX) model. We present this scheme in the context of Greedy routing [16] and elaborate how AR and ARX are adopted to manage trust. According to the proposed scheme, a node a periodically measures the packet forwarding ratio of its every neighbor b. The measurements are the trust observations about b; they form a time series of data. Then, node a applies an autoregression model to the time series and predict b's future packet forwarding ratio as the trust es-

timate (or simply trust value) of b. In routing, a takes b as next hop if b is the one with the high trust value closest to destination among all neighbors. With an AR model being applied, a only uses its own observations to predict b's trust value; with an ARX model, a also takes into account the collective recommendation from neighbors, which is the weighted average of the trust values of b in those neighbors opinion. Through extensive simulation, we evaluate the scheme's performance in comparison with an existing trust management scheme based on Bayesian model [17]. Our simulation results indicate that use of the ARX model leads to the best trust estimation accuracy and the largest improvement on routing reliability.

Part of this thesis has been accepted for publication at the IEEE Global Communication Conference (Globecom), Texas, Dec. 2011 [27].

1.3 Thesis Organization

The thesis is organized as follows. Chapter 2 describes the background and related work. Chapter 3 introduces the basic knowledge, with mathematical perspective, of AR and ARX model. The details of how these two models would work in a trust management system are discussed in Chapter 4. Chapter 5 presents the simulation analysis and results. The conclusion and future work is presented in Chapter 6.

Chapter 2

Literature Review

Trust management (TM) foundations are introduced in this chapter. Specifically, TM approaches for wireless ad hoc networks will be discussed. Then, the models for trust management systems classification will be outlined.

2.1 Trust Management

In wireless ad hoc networks, threats occur in many transactions. Nodes need to question if they can relay the messages or services received from other peers and also if they can trust them. Trust management (TM) [7] is a mechanism that assigns each node a trust status, based on long-term observations, in order to help judge the degree of trustworthiness of peers. A TM scheme aids each node monitor and predict the behavior of its neighbors.

2.1.1 Trust

In different fields such as psychology, sociology, law and business, the definition of trust is captured by dissimilar standpoints. According to the Merriam-Webster dictionary [47], trust is defined as "*assured reliance on the character, ability, strength, or truth of*

someone or something".

Originally, trust was used in a social context to represent relationships among people. Then, as networks started to resemble human life, trust was introduced into this field to solve the confidence problems mirroring those that occurred in the social world. Trust in wireless ad hoc networks is derived from the definition pulled from social settings. Various applications in wireless ad hoc networks give rise to different conceptions of trust. In this chapter, we prefer to follow the definition proposed by Eschenauer et al. [15], that is: "*a set of relations among entities that participate in a protocol. These relations are based on the evidence generated by the previous interactions of entities within a protocol. In general, if the interactions have been faithful to the protocol, then trust will accumulate between these entities*" [15].

Trust in wireless ad hoc networks exhibits the following features:

Non-Transitive If A trusts B and B trusts C, it cannot simply be inferred that A trusts C.

Non-Static If A trusted B in the past, it cannot imply that A will always trust B.

Non-Mutual If A trusts B, it doesn't mean B also trusts A.

The trust relationship between two peers can help determine whether A trusts B or not. It can't describe the degree to which a peer can be trusted. An indicator is needed to show the quality of an entity's service. Trustworthiness, sometimes expressed as a trust value, can be used as an indicator of the probability of a peer's future behavior. The computation of the trust value is important in building reliance within a network. Trust management systems are studied with the goal of establishing a computational model to solve the trust problems in peer-to-peer cooperation.

2.1.2 Trust Management Systems

Trust management was first proposed by Blaze [7] in 1996 as a coherent framework for the study of security policies, security credentials and trust relationships [7]. Security policies refer to the trust assertions provided by local systems. Security credentials are the signed trust certificates for entities of a local system which are evidence of their identities. Trust relationships are the relationships based on these policies and credentials. For example, consider an electronic banking system. At least k employees can approve loans of a large amount of money (security policies). The bank needs to enable officers with evidence to prove they are members included in these k employees (security credentials). Furthermore, the bank makes decisions as to who can issue services (trust relationship) [7].

PolicyMaker [9] and KeyNote [8] are the earliest trust management systems. Basically, they all use credential verification to establish the trust relationships among peers. PolicyMaker [9] was proposed by Blaze and works like a central database which can combine the nodes' local policies and credentials with the evaluated trust actions and return a decision on whether or not to allow the assessed action. KeyNote [8] provides a unified language to describe security policies, security credentials and trust relationships. When a principal (i.e., an entity authorized to perform actions) requests an action, the system uses KeyNote to correctly implement the security policies, security credentials and trust relationships, and then KeyNote returns a Policy Compliance Value to the system which shows how to deal with the request.

With researchers continuously studying trust management, there is an abundance of such systems introduced to solve reliability issues in wireless ad hoc networks. They can be categorized into two major branches: Credential-Based Trust Management Systems and Reputation-Based Trust Management Systems.

2.1.3 Credential-Based Trust Management Systems

In this type of systems, the trust relationships among entities (established by managing and exchanging credentials) is verified and restricted by a preset policy. An entity will be trusted after its credentials have been verified. Every trusted entity can issue its credential to an unknown entity to become trusted. Credentials, for now, are broadly applied as access rights in peer-to-peer (P2P) environment networks. They are always used for establishing trust relations among the participating entities [44].

The early PolicyMaker and KeyNote TM systems are both credential-based. Public Key Infrastructure (PKI) [41] is another seminal system that has spawned many extensions or modifications. PKI is a set of hardware, software, people, policies and procedures needed to create, manage, distribute, use, store, and revoke digital certificates [40]. In P2P networks, PKI can be utilized as an arrangement combining public keys and an entity's identity together, based on a certificate authority (CA). The CA is a trusted center. In some projects, a trusted third party is used to verify, issue and revoke certificates which can delegate CA and grant rights to end users. The credentials in trust management here are regarded as certificates in a PKI system.

Although credential-based TM systems can solve the confidence issues that arise in P2P networks, they still have limitations. For example, they usually make binary decisions on whether to allow the request or not based on whether the requestor is trustworthy or not. Because of this binary modeling, it seems that credential-based TM systems lack flexibility. On the other hand, reputation-based TM systems perform better and focus on evaluating the degree of trust.

2.1.4 Reputation-Based Trust Management Systems

Reputation is being widely used in various subjects such as psychology, sociology and business. In the Merriam-Webster dictionary, reputation is defined as "*the beliefs or opinions that are generally held about someone or something*" [47]. Reputation is one

important mechanism that people employ to build trust relationships in wireless ad hoc networks, since a peer's reputation is based on feedback from other peers who have had direct cooperation with it. This feedback is set to represent how trustworthy a peer performs during cooperations. Given a set of feedbacks, reputation can be computed by means of trust functions and represents a peer's degree of trustworthiness, usually ranging from 0 to 1, where 1 means the peer is trustful and 0 indicates that the peer is trustless.

In recent years, reputation-based trust management systems have been amply studied. Any such a system calculates the reputation value of a node via gathering observations of the node's behaviors. It does not need any historical experience or prior reputation to measure trustworthiness. Rather, it can set its initial trust value at the beginning. When peers in the system establish a trust relationship with another peer, a trust value is always assigned to this relationship and often computed through reputation during a period of time, that is, Trust value=Φ{Reputation, the Time elapsed since the reputation was last modified} [4], where $\Phi\{\}$ is the trust function used to compute the trust values. In order to select the trustful peers, reputation-based systems are sometimes equipped with a threshold to eliminate the malicious and selfish nodes which usually have low trust values, or a set range from 0 to 1 for representing the trust status of a peer.

In [34], a reputation-based TM protocol for P2P networks was proposed. It is used to distinguish malicious nodes from benign nodes on the basis of utilization, rate and reliability. The scheme assigns to every peer a trust vector to maintain the outcomes of the past interactions with other peers. After every transaction, the responses are sorted and weighted by the credibility rating of the responder. The credibility rating is similar to the trust vector, with a value set between 0 and 1, where the two endpoints represent the failure or success of past transactions, respectively. If peer A wants to issue a trust query to peer B, then the trust value of a peer is described as $\frac{\sum_{i=1}^{k} c_i t_i}{k}$,

where c_i is the credibility rating of A, t_i is the trust rating of B for A and k is the number of times they have interacted in the past. So after this process concludes, B has to judge whether A is malicious or benign according to the trust value.

Reputation-based TM systems can help build trust relationships among peers by computing trust values. However, they still have a few drawbacks due to their inherent nature of relying upon the feedback of other peers whose trust values can not be calculated accurately. They may also be dependent on manual settings and thus lack self-adaptive qualities. To solve this problem, researchers start to combine reputation and credentials when creating trust management systems. In [18], the authors employ three modules to make up the trust system, i.e. reputation module, credential learning module and integration module. The reputation modules exploits the Bayesian approach [23] to calculate a peer's reputation based on the feedback from all other peers that have interacted with it. The credential learning module computes the trust values of the peers who have credentials. Finally, the integration module derives the overall trust value by combining reputation and credential values. In this cooperative way, the trust building process is more accurate and adaptive.

2.2 Trust Management Models

In general, a TM system leans upon three schemes during the whole trust building process: the trust collection scheme, the trust computation scheme and the decision making scheme. The trust collection scheme is used to gather information on the peers based on their behaviors, which is the foundational aspect in TM systems. It provides the trust computation scheme with the data source to construct the trust relationship. The trust computation scheme calculates the trust values by means of various mathematical techniques that are applied to the data supplied by the trust collection scheme. Afterwards, the final decision should emerge from the decision

making scheme, which is designed to evaluate the results from the trust computation scheme and has the responsibility of determining which peers should be trusted or selected. Usually, a threshold mechanism is used to filter untrustworthy peers, but setting a level of trust is an optional efficient way to choose peers. In this section, we will explore the literature to show how these three schemes work in TM systems in presence of dissimilar models.

2.2.1 Game Theory for Modeling Trust

Game theory is a branch of applied mathematics popularized by J. V. Neumann [31]. "*Game theory is a sort of umbrella or 'unified field' theory for the rational side of social science, where 'social' is interpreted broadly so as to include human as well as non-human players (computers, animals, plants)*" [5]. It has been extensively used in wired and wireless networks for various types of works, such as power control and trust management. There are two types of games, i.e. cooperative and non-cooperative. In the former, the nodes' choices to maximize their payoffs depends on the choices of others in a competition whereas in the latter the nodes make their own choices independently.

Generally, a *game* is a model that can address the interaction issues among the *players* who take actions (make decisions) based on some strategies, and there are *payoffs* assigned for each combination of strategies. Game theory comes to play an increasingly important role in modeling wireless ad hoc networks and several models have been introduced to manage trust problems therein. These models are studied using various kinds of game strategies but in this chapter they will be simply discussed from their cooperation perspective, i.e. cooperative models versus non-cooperative models.

A cooperative game in [35] aims to build a trust computation model where nodes interact with their neighbors locally. First, it should assign every node a trust level

value. For example, if node B wants to verify node A to see how trustworthy A is, B should evaluate the packet forwarding rate of node A and then, using Figure 2.1, lookup the trust value of node B in node A.

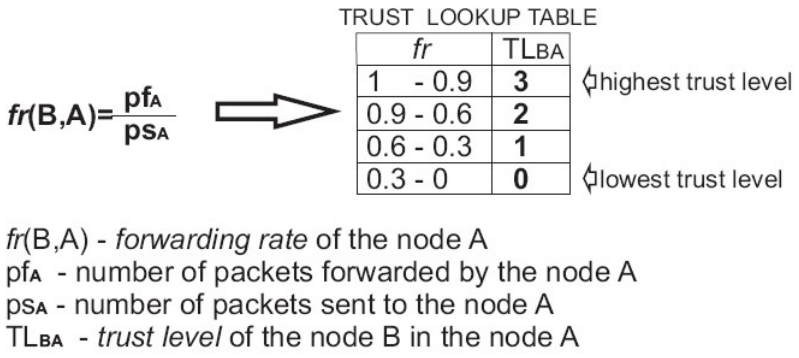

Figure 2.1: Trust value lookup [35]

Consider a network where there is a source node A that wants to send a packet to node D by using the intermediate nodes B and C. So the game participants are nodes A, B and C. After node B receives the packet from A, B has two options concerning this packet, i.e. to forward it or to discard it. If B chooses the latter, the game ends and every participant node will receive payoffs according to their decisions made during the game. There are two types of payoffs: one assigned to the source nodes and the other to the intermediate nodes. For source node payoffs, if the packet reaches the destination node, then the payoff will be set to "S" which represents *success*, otherwise it will be set to "F" if the transmission has failed. Payoffs for intermediate nodes are contingent upon the nodes' decisions and the trust values the source nodes have for them. Generally, the intermediate nodes with higher trust values will always receive high payoffs when they forward the packets. Nodes with higher trust levels are more likely to transfer the packet than nodes with low trust levels concerning their future behavior. If a node discards the packet, then it can be rewarded for saving its battery but its reputation will suffer. So an untrustful node will receive a better payoff when it discards a packet coming from a poorly trusted node and the game may not profit in

building trust among game participants. An example is shown in Figure 2.2. A wants to send packets to D using intermediate nodes B and C. The result for this game is failure because C discards the packet (Figure 2.2b). After the game, the payoff assigned to intermediate node B is $TL3$ because it sent the packet and C received an appropriate payoff $TL1$ due to its dropping the packet, as shown in Figure 2.2. Then the final payoff for source node A is "F" because the transmission has failed.

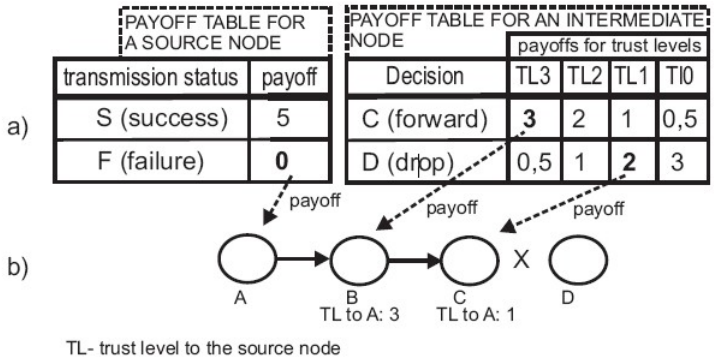

Figure 2.2: Payoff for source and intermediate nodes [35]

A non-cooperative game was developed in [22] between two players. It is known as *The Prisoners Dilemma* and posed in the network without full cooperation. The nodes in this network are assumed to be selfish but not malicious, that is, they want to maximize their own profit at the minimum cost of their actions. To model the two-player game in this paper, it iteratively repeats the game that assumes that nodes send packets to each other and then decide on whether to forward or drop them. There is a reward for each node that chooses to forward the packets and it is set to α, where $\alpha \geq 1$. The payoff matrix (in its normalized version) is depicted in Figure 2.3. Given the probability of a node forwarding a packet, the average payoff at a time slot can be computed. With this game, the present paper used a reputation strategy named DARWIN to establish a full cooperation between the selfish nodes. It assumes that a player is always in good standing on the first stage and remains like that if her dropping probability is less than the predefined DARWIN dropping probability. Otherwise, it

turns to a bad standing. If a bad standing player's opponent is in good standing, this bad standing player still could cooperate, otherwise it will defect. DARWIN proposes a measurement of the standing statuses and assumes that players in this network could share the measurements so that the misbehaved and selfish players cannot lie about their standing statuses. This way, the selfish players have limited impact on establishing the cooperative network.

Payoff Matrix of the Packet Forwarding Game

		Node *j*	
		Forward	Drop
Node *i*	Forward	α-1 α-1	-α-1 α
	Drop	α -α-1	-α -α

Normalized Payoff Matrix of the Packet Forwarding Game

		Node *j*	
		Forward	Drop
Node *i*	Forward	1 1	$\frac{-1}{2\alpha-1}$ $\frac{2\alpha}{2\alpha-1}$
	Drop	$\frac{2\alpha}{2\alpha-1}$ $\frac{-1}{2\alpha-1}$	0 0

Figure 2.3: Normalized Payoff Matrix of the Packet Forwarding Game

2.2.2 Graph Theory for Modeling Trust

Graph theory studies mathematical structures that are a collection of vertices and edges connecting pairs of these vertices [46]. When building trust in wireless ad hoc networks, graph theory is applied to analyze the local interactions among peers.

The study of trust problems in [39] describes a directed graph problem. The peers in the network do not have enough information about each other. In this graph, the vertices represent the peers and the edges stand for their trust relations. The edges are

weighted by the opinions from a peer about another peer. Each opinion consists of two items: the trust value and the confidence value. The former is the issuer's estimation of the target peer's trustworthiness whereas the latter denotes how accurate the former is. If a peer has conveyed most of the messages the issuer has sent or it has not acted maliciously during its interactions with the issuer, then the confidence value on the issuer's opinion is high. A higher confidence value makes the opinion more useful. Both the trust value and confidence value are assigned by the issuer to the target peer based on local observations using the issuer's own criteria.

There are two versions of the trust inference problem. The first is to find the confidence value and trust value from peer A on node B. It can be viewed as the generalized shortest path problem, i.e. the problem can be described as finding the general distance between A and B. The second version is finding the most trusted path between A and B, which is the path with the highest aggregate trust value among all the trust paths from A to B.

2.2.3 Bayesian Theory for Modeling Trust

A Bayesian model [46] is a probabilistic model that relies on a Beta probability density function $Beta(\alpha, \beta)$ represented as:

$$f(p|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{2.1}$$

where p is the probability of the event occurrences and should be in the range [0,1], and $\beta > 0$, $\alpha > 0$. A Bayesian model can be applied to analyze trust management systems for wireless ad hoc networks. Usually, when using a Bayesian model to analyze reputation information, one employs two kinds of information: first-hand information and second-hand information. The former is in general obtained by direct observation of any two entities after they have cooperated with each other. Second-hand informa-

tion can be defined as the opinions of the other entities about the said participating entities.

If we assume that in a reputation-based TM system there are r successful transactions and s failed transactions, then in a Bayesian model α in (2.1) will equal to $r+1$ where $\beta = s+1$. Authors in [29] discussed the Bayesian model built on second-hand information. For each node, there are other nodes that can carry its reputation. For example, R_{ij} is the reputation of node j carried by a neighboring node i. α_{ij} is the number of successful transactions between node i and node j while β_{ij} is the number of failed transactions between them. Then, the reputation of node j carried by node i is $R_{ij} = Beta(\alpha+1, \beta+1)$. For a more precise evaluation, second-hand information is added to this model. The second-hand information provided by node k to node i about node j is denoted as R_{kj} with two parameters $\alpha_{kj} and \beta_{kj}$. By amalgamating these new pieces of information, a new reputation for node j is defined as $R_{ij}^{new} = Beta(\alpha^{new}, \beta^{new})$. The expected value of reputation T_{ij} can be derived as $T_{ij} = E(R_{ij}^{new})$. The expected value T_{ij} could help distinguish misbehaving nodes from normal nodes.

In [43], a modified Bayesian scheme for a reputation-based TM system is put forth. It rates reputation and trust based on first-hand reputation information. In this model, the reputation rating from node i to node j is expressed as $R_{i,j}$ according to its first-hand observations based on past cooperation experiences. The trust rating $T_{i,j}$ is node i's opinion about node j on how honest j performs in the system. The first-hand information $F_{i,j}$ is the summary opinion that i has about j. In the reputation rating of this model, node i uses θ to represent the probability of node j's misbehavior, which is updated when first-hand information changes. F_{ij} has a $Beta$ parametric form (α_n, β_n) with initial value (α_0,β_0) set to (1,1). A weight u is introduced in this reputation rating formula as a fading mechanism because past experience should have less influence on the new reputation. So, the values of α_n and β_n after n first-hand observations are:

$$\alpha_n = s_n + us_{n-1} + u^2 s_{n-2} + ... + u^{n-1} s_1 + u^n \tag{2.2}$$

$$\beta_n = u^n + u^{n-1} + ... + 1 - u^{n-1} s_1 - u^{n-2} s_2 - ... - us_{n-1} - s_n \tag{2.3}$$

where s_1, s_2,...,s_n is the sequence of observations. When $s_i = 1$ the observation is regarded as misbehavior, otherwise $s_i = 0$.

The reputation value R_{ij} will be updated in the same way that first-hand information F_{ij} is. In trust rating, ϕ is computed using a similar Bayesian approach to that in F_{ij} in order to derive the probability of node k giving false reports to node i. T_{ij} is updated as long as there is first-hand information F_{ij} for node i from node k about node j. After updating the reputation value and trust value, the model utilizes two thresholds, t_r and t_u, to help node i classify the behavior and trustworthiness of node j. When node j's $\theta \geq t_r$, node j is tagged as misbehaving, otherwise it is normal. If its $\phi \geq t_u$, it is regarded to be untrustworthy, otherwise it is trusted by node i. Central to this model is the thresholding method employed in the decision making scheme.

2.2.3.1 BRSN Model

In order to analyze the performance of the autoregression model family, we need another class of models to compare them to. Bayesian models are good choices because they have been studied for a long time and can help the trust system efficiently build a reliance relationship. In this section, we briefly introduce the Bayesian system and discuss the Beta Reputation system for Sensor Networks model (BRSN) [17] in detail.

BRSN is the work of Ganeriwal and Srivastava. It gives a reputation rating based on both first-hand information and second-hand information, which helps avoid a subjective rating. It also provides an aging parameter that gives less weight to historical information.

In BRSN, a node n_i holds a reputation value for its neighbor node n_j, which can be represented by a Beta model as $R_{ij} = Beta(\alpha_{ij} + 1, \beta_{ij} + 1)$. α_{ij} and β_{ij} are the parameters node n_i has about node n_j. They are all non-negative real numbers. The prediction of trust is the expectation value of the reputation, expressed as follows:

$$T_{ij} = E(R_{ij}) = \frac{\alpha + 1}{\alpha + \beta + 2} \tag{2.4}$$

For updating the reputation, BRSN runs $r + s$ more events, where r represents the number of cooperative transactions, and s represents the number of non-cooperative transactions. Then α and β can be calculated by r and s respectively as shown below:

$$\alpha = r + 1; \beta = s + 1 \tag{2.5}$$

When a transaction is cooperative, then r will be augmented by 1, otherwise s will increase by 1. The information of r and s is provided by a watchdog block in the system. This block is used for collecting observations and making decisions. The update of the Bayesian system is straightforward because it only requires updates for r and s after each transaction.

Recent information should be given more importance and historical information should be attenuated by lowering its weight. This is achieved in the following way:

$$\alpha_{ij}^{new} = (\omega_{age} \times \alpha_{ij}) + r; \beta_{ij}^{new} = (\omega_{age} \times \beta_{ij}) + s; \tag{2.6}$$

where ω_{age} is the aging weight (randomly ranged between 0 and 1). The weight selection is important in modeling reputation information.

First-hand information alone is not sufficient. Here, BRSN included second-hand information provided by node n_i's neighbor, n_k, about node n_j, which is R_{kj}. Node n_k has parameters (α_{kj},β_{kj}) about node n_j. The combination can be derived as:

$$\alpha_{ij}^{new} = \alpha_{ij} + \frac{2 \times \alpha_{ik} \times \alpha_{kj}}{(\beta_{ik} + 2) \times (\alpha_{kj} + \beta_{kj} + 2) + 2 \times \alpha_{ik}} \tag{2.7}$$

$$\beta_{ij}^{new} = \beta_{ij} + \frac{2 \times \alpha_{ik} \times \alpha_{kj}}{(\beta_{ik} + 2) \times (\alpha_{kj} + \beta_{kj} + 2) + 2 \times \alpha_{ik}} \tag{2.8}$$

where α_{ik} and β_{ik} are the parameters that node n_i has about node n_k. Here, n_k's second-hand information is weighted by its reputation. In this case, a more precise prediction can be accomplished.

With the second-hand information, the new reputation can be turned to:

$$R_{ij}^{new} = Beta(\alpha_{ij}^{new}, \beta_{ij}^{new}) \tag{2.9}$$

2.2.4 Markov Chain for Modeling Trust

The Markov chain is a random process whose future status is determined by its current status [46]. So, it can be used in trust management systems to predict the trust value.

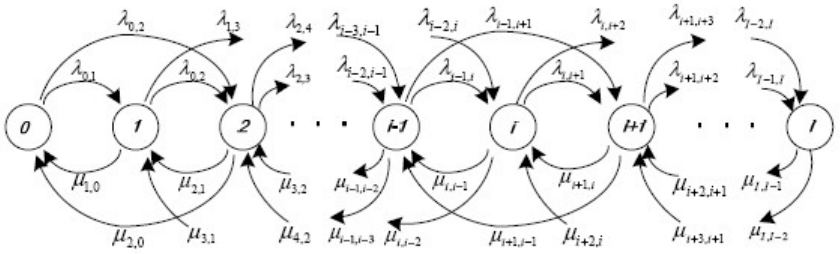

Figure 2.4: State diagram of the trust state of the proposed approach [13]

A trust derivation approach based on Markov chain for the Multicast MANET is put forward in [13]. There are two main phases in this model. One is used to determine the trust value using Markov chain analysis and the other is central authentication (CLA) selection based on the trust value obtained in the first phase, which can be envisioned as a threefold process. Initially, trust relations among peers are created. Each node in the network evaluates the trust value of its one-hop neighbors. Then, the node with

Table 2.1: Trust state table

Good Manners	Trust Value	Bad manners	Trust value
Normal leaving	+1	Abnormal leaving	-1
Normal joining	+1	Abnormal joining	-1
Available power is greater than a higher bound	+1	Available power is less than a lower bound	-1
Available bandwidth is greater than a higher bound	+1	Available bandwidth is lower than a lower bound	-1
Win the BCA competition and it is not the current CLA or BCA	+1	Lose the BCA competition and it is the current CLA or BCA	-1
Win the CLA competition and it is not the current CLA	+2	Lose the CLA competition and it is the current CLA	-2

highest trust value is selected as CLA and that with the second highest trust value as the backup CLA (BCA). The second step is to introduce events for the transitions among trust states in the Markov chain. A peer's trust value will change if the trust state changes. The trust state is assumed in this paper for analyzing the steady state of each peer's trust value. The trust state is shown in Table 2.1, where the trust value with good manners can increase but the ones with bad manners will decrease. For example, a node is inferred to be malicious by sending many *JOIN* messages in a short time, thus attempting to intrude in the network. An *abnormal joining* like this should be regarded as malicious activity. Figure 2.4 displays the trust state using the Markov chain model, in which the $\lambda_{i,i+1}$ and $\lambda_{i,i+2}$ are the arrival rates at state i, and $\mu_{i,i-1}$ and $\mu_{i,i-2}$ are the departure rates at state i. State 0 represents the lowest trust state and state I is the highest trust state. If a peer acts in a good manner listed in the Figure 2.1, then the state will transition from low-trust state to high-trust state, otherwise it will move from high-trust state to low-trust state. The final step is to determine the trust value by calculating the steady trust state probability of each peer using the

Markov chain. Then with the final trust value, which is related to its historical trust states, the network can efficiently filter the malicious peers to avoid the intrusion and pick a CLA to solve the authentication problems in distributed MANETs.

The confidence problem in trust management systems is complicated and usually requires one or more approaches working together. Though trust management systems with Markov chain model are good at predicting the trust states of peers, they still need an efficient approach to evaluate trust values to help with these predictions. Authors in [28] proposed a system putting the Bayesian approach and the Markov chain approach together to prevent good peers from transacting and cooperating with the malicious peers in wireless ad hoc networks.

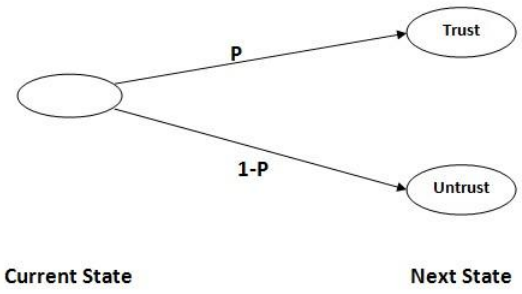

Figure 2.5: The State transition diagram

The Bayesian approach aims at calculating the trust value in a more precise way. It collects the historical reputation feedbacks in the system and extracts the Bayesian coefficients using a certain extraction rule. Then, it uses the reputation feedbacks and Bayesian coefficients as inputs to the reputation algorithm that computes the trust values.

The Markov chain approach analyzes how the future trust state is influenced by the current trust state. In Figure 2.5, p represents the probability of a peer behaving in a trustworthy fashion in the next state. This indicator differs across various P2P pairs. For example, the $p(a, b)$ between peers a and b is different from $p(b, c)$ between

peers b and c because they are dependent on their previous transaction experiences. With the probability of these P2P pairs, one can create a generator matrix that can be adjusted by the Bayesian coefficients to predict the trust value. In order to predict the trust state using Markov chains, one needs to establish an available initial Markov state. In this paper, it is assumed that the peers are trustworthy at the beginning, at which time they are called pre-trusted peers. The next step is to find the trust value of the steady state, which can be computed via a reputation algorithm proposed in this paper and expressed as:

$$\overrightarrow{T_k^T} = \alpha C \overrightarrow{T_{k-1}^T} + \beta \overrightarrow{P_0^T} \qquad (k \geq 1) \tag{2.10}$$

$$\overrightarrow{T_k^T} = \overrightarrow{P_0^T} \tag{2.11}$$

where α is the generator matrix weight, β is the pre-trusted peers' weight, C is the transition matrix, P_0^T is the initial generator matrix, and T_k^T is the trust value we want to predict. When the difference between T_k^T and T_{k-1}^T is less than a very small number ε, then we believe that T_{k+1}^T reaches the steady state and $T(i)_k$ is considered as the final prediction trust value of peer i. Based on the predicted trust value, nodes can distinguish the malicious peers from the good peers and may finally decide which peer to interact with. Therefore, this model aids the network avoiding intrusion by malicious peers.

2.2.5 Fuzzy Logic for Modeling Trust

Fuzzy logic is derived from fuzzy set theory to deal with approximate rather than accurate multiple trust values [46]. It is implemented in a trust management system by means of a predefined fuzzy set. A fuzzy set is a set in which its elements have their degrees of membership to the underlying concept described through membership functions [46]. In trust management, this mathematical fuzzy set of the trust values

Table 2.2: The group of fuzzy sets

Level	Linguistic Variable	Signification
L1	Complete	Completely trust
L2	Good	More trustworthy than most entities
L3	Average	Mean trustworthiness
L4	Minimal	Lowest possible trust
L5	Ignorance	Can not make trust-related judgement
L6	Distrust	Completely untrust

should be first defined. Afterwards, fuzzy operators are employed to obtain the fuzzy result in order to solve the confidence problem in TM systems.

An example was given in [45]. The fuzzy set was defined as $A = (x, A(x))$ where x belongs to the set of X, which is either the source of the information or the entity which builds the trust relationship, and $A(x)$ represents the membership function of every x. To establish a real trust relationship, there exist different types of uncertainty which are categorized into 6 groups in this paper, as portrayed in Table 2.2. To determine the degree of uncertainty, two kinds of trust need to be calculated. One is the direct trust, represented as $a \longrightarrow^{D} b[L]$, where entity a is willing to rely on entity b to a degree D under uncertainty L. The other one is the recommendation trust, defined as $a \longrightarrow_{R}^{D} b[L]$, which infers that entity a is willing to rely on entity b to a degree D with the recommendation R under uncertainty L.

The calculation of the degrees for capturing different types of uncertainty follows after deduction and consensus rules. The deduction rules work in a transitive manner, i.e. they transfer trust from entity to entity in order to create a recommendation path. For example, if entity i wants the recommendation of j through k_1, then the recommendation path will be $j \longrightarrow k_1 \longrightarrow i$. If there exists two or more recommendation paths, then the consensus rules are used to merge the multiple recommendations into a singleton comprehensive recommendation. In this process, the calculation of the degree D and the uncertainty L can be determined. This model proposed a general sense in modeling trust using fuzzy logic.

Table 2.3: Inference rules for trust aggregation

Rule #	Description
1	If $(t - T_{zy}((t), c))$ is R and $TF_{zy}(c)$ is H, then ω_i is VH
2	If $(t - T_{zy}((t), c))$ is R and $TF_{zy}(c)$ is M, then ω_i is H
3	If $(t - T_{zy}((t), c))$ is R and $TF_{zy}(c)$ is L, then ω_i is M
4	If $(t - T_{zy}((t), c))$ is O and $TF_{zy}(c)$ is H, then ω_i is H
5	If $(t - T_{zy}((t), c))$ is O and $TF_{zy}(c)$ is M, then ω_i is M
6	If $(t - T_{zy}((t), c))$ is O and $TF_{zy}(c)$ is L, then ω_i is L
7	If $(t - T_{zy}((t), c))$ is VO and $TF_{zy}(c)$ is H, then ω_i is M
8	If $(t - T_{zy}((t), c))$ is VO and $TF_{zy}(c)$ is M, then ω_i is L
9	If $(t - T_{zy}((t), c))$ is VO and $TF_{zy}(c)$ is L, then ω_i is VL
10	If a given trust level is UNKNOWN,ω_i is zero

Table 2.4: Description of the fuzzy trust levels

Trust Level	Description	TFN
VL	very untrustworthy	$[-1.25, 0, 1.25]$
L	untrustworthy	$[0, 1.25, 2.5]$
M	medium trustworthy	$[1.25, 2.5, 3.75]$
H	trustworthy	$[2.5, 3.75, 5]$
VH	very trustworthy	$[3.75, 5, 6.25]$
U	unknown	$[0, 0, 0]$

In [6], a fuzzy model to weigh the direct trust and reputation (second-hand information) in a P2P-based system was presented. The linguistic labels are defined to help peers assign the trust level intervals instead of the exact trust values. The linguistic label values range from very untrustworthy to very trustworthy and are represented in Table 2.4, where a TFN (triangular fuzzy number) for each trust level is defined. The TFN specifies the range of every trust level rather than a specific value. The truth of TFN having only two parameters known as center point and width makes the TFN calculation simple. The trust level that peer x holds towards peer y is computed by taking into account the direct trust (which is contingent on the direct interaction experience between x and y) and the reputation that is shaped by interactions with third party trusted peers, for example, peer z. Both direct trust and reputation are weighted

by ω_0 and ω_i respectively. For reputation weighting ω_i calculations, fuzzy inference rules are introduced on the basis of the recommendation time stamp $t - \tau_{zy}((t), c)$ and transaction frequency $TF_{zy}(c)$ for context c reported in Table 2.3, where for each recommendation $t - \tau_{zy}((t), c)$ fuzzified by the membership function in Figure 2.6, there are three fuzzy sets: *recent(R), old(O) and very old(VO)*, and for each $TF_{zy}(c)$ fuzzified, there are also three fuzzy sets: *low(L), medium(M) and high(H)* defined by the membership functions depicted in Figure 2.7.

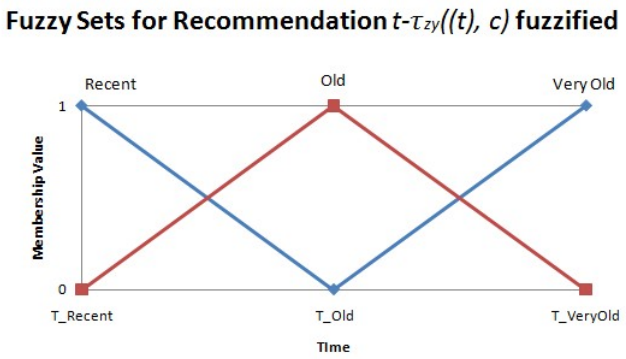

Figure 2.6: Recommendation age membership functions

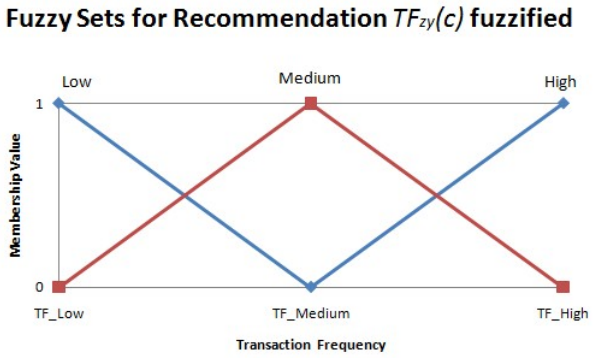

Figure 2.7: Transaction frequency membership functions

The final trust level can be computed with these weights. This model is rooted on fuzzy logic to determine the weights of direct trust and reputation and then to yield the final trust level. It is a flexible model because the weights can be adjusted for different trust systems.

Chapter 3

Proposed Model for Trust Management

For modeling trust management systems, we prefer to use the autoregressive model with exogenous inputs. There are a lot of models related to autoregression and used to analyze time series. Autoregression is always represented by autoregressive models in which the dependent value is the current value and the independent values are the previous values from the time series. We will review some predominant models to generate a macroscopic view about the autoregression. With this background in mind, we can better understand the model being brought forward in this chapter.

3.1 Autoregressive (AR) Model

The autoregressive model (AR) [14] is designed for analyzing and forecasting time series such as average daily temperature or earthquake magnitude over the years. AR predicts the output value $y(t)$ based on its p previous values $y(t-1)$,$y(t-2)$,...,$y(t-p)$. A p-order autoregressive model, denoted by AR(p), can be expressed as follows:

$$y(t) = c + \sum_{i=1}^{p} \varphi_i y(t-i) + \varepsilon(t) \tag{3.1}$$

where c is a constant, $\varepsilon(t)$ is the white noise term with zero mean, $y(t)$ represents the output at time t, φ_1,...,φ_p are the model coefficients and p is the order of the model. The constant term is often omitted by many authors for simplicity.

Time series are either stationary or non-stationary. In the autoregressive model formula (Equation 3.1), if $|\varphi| < 1$, $y(t)$ it is a stationary time series, otherwise the time series is not stationary [14]. The AR model is used when the series is stationary.

In Equation 3.1, the output y at time t can be estimated as long as the parameters φ_i and the order p of the model are known. The problem with AR model analysis is how to get the proper parameters and order that best represent the model. A number of algorithms have been developed to calculate AR model's parameters. The Least-Square (LS) [38] and Yule-Walker (YW) [38] approaches are the basic methods most commonly used. The vast majority of the remaining algorithms are modifications or improvements of them. The cycle generalized least squares [42] approach, for example, is a modification of LS focused on improving the efficiency in AR model simulation. There is also a modified Yule-Walker equation algorithm [26] geared at estimating noisy AR processes more accurately. LS is employed to select the best parameters when they can minimize the squared residuals. A residual is the difference between the observed values and those synthesized by the model.

In the AR inference process, let Θ denote the unknown parameter vector:

$$\Theta^T = [\varphi_1, \varphi_2, ..., \varphi_p] \tag{3.2}$$

By using LS to estimate the parameters, one arrives at the equation described below [38]:

$$\hat{\Theta} = \arg\min \sum_{t=2}^{N} [y(t) - \Phi^T \Theta]^2 \tag{3.3}$$

where N is the number of elements in the series. If the order is determined in advance, then N will be the order p and the data vector Φ is:

$$\Phi = [-y(n), -y(n-1), ..., -y(1)]^T \tag{3.4}$$

After the calculations, we can obtain the parameters Θ.

Another element needed to determine an AR model is the order, i.e. the number of the parameters in the AR model. Since it is a crucial part in AR model processing, many order selection approaches have been proposed due to the fact that the AR model has been exploited in various types of systems modeling particularly in natural and social environments, e.g. wind speed forecasting [20] and electrical brain activity mapping [32].

3.2 Autoregressive with Exogenous Inputs (ARX) Model

The ARX model is an extension of AR with inputs representing external information. An ARX model with p outputs and b inputs can be denoted by $ARX(p, q)$. Its equation is written as follows:

$$y(t) = \varepsilon(t) + \sum_{i=1}^{p} \varphi_i y(t-i) + \sum_{i=1}^{b} \eta_i d(t-i) \tag{3.5}$$

where φ_1, φ_2,...,φ_p are the parameters of outputs $y(t-1)$,...,$y(t-p)$, and η_1, η_2,...,η_b are parameters of inputs $d(t-1)$,...,$d(t-1)$.

To determine the order of an ARX model, we can also use the AIC method. To estimate parameters, it often runs LS to find the minimum-error sum of squares. In ARX parameter estimation, the parameter vector and data vector will change to:

$$\Theta^T = [\varphi_1, \varphi_2, ..., \varphi_p, \eta_1, \eta_2, ..., \eta_b] \tag{3.6}$$

$$\Phi^T = [y(t-1), y(t-2), ..., y(t-p), d(t-1), d(t-2), ..., d(t-b)] \tag{3.7}$$

Another way to compute the parameters is via the maximum likelihood method analyzed and popularized by R. A. Fisher [3]. The maximum likelihood method intends to maximize the likelihood function. In ARX models, such function will be represented by [25]:

$$Ł(\Phi^T|\Theta^T) = \prod_{i=1}^{N} f(\Phi^T; \Theta^T) \tag{3.8}$$

where N is the number of the estimated data samples, $f(\ )$ is the probability density function of data noise—modeling error vector. The optimal parameters are selected by maximizing the value of Ł.

The LS method and maximum likelihood method are equivalent in terms of parameter estimation.

ARX can serve to create an automatic stock market forecasting and portfolio selection mechanism [21]. In this study, the financial data are collected automatically every quarter and furnished as input to an ARX prediction model so as to forecast their future trends over the next quarter or half-year period.

3.3 Measuring Model Accuracy

For evaluating the model accuracy we need some criterion, which we will discuss in this section. The first popular approaches were Final Prediction Error (FPE) [1, 2] and Akaike information criterion (AIC) [2].

FPE is the mean square error of the one-step-ahead prediction based on the LS-estimated parameters of the AR model [2] and it is defined as:

$$FPE(p) = \frac{N+p}{N-p} P_p \tag{3.9}$$

where P_p is the residual squared error for a p-order AR model and N is the number of the estimated data samples. We take the order p when $FPE(p)$ is minimized. In this case, we choose the order that minimizes the model's average error given the AR parameters.

AIC stands for an information criterion, which is a measure of the goodness of the model's fit [2]. In general, AIC is represented by:

$$AIC(p) = \ln P_p + \frac{2p}{N} \tag{3.10}$$

Again, we select the order p when $AIC(p)$ is minimum. In this way, we find the best model order to represent the data samples with the smallest number of parameters.

We also make use of the AIC and FPE values to evaluate how well a model works for a certain system. The model with the minimal AIC and FPE value is the best one. That is to say that the best model we select should efficiently represent the system and also have a minimum final prediction error.

There are other widespread approaches to derive the model order, including the Bayesian information criterion (BIC) [33], Kullback information criterion (KIC) [12] and the minimum description length (MDL) [19].

Chapter 4

Customizing AR and ARX Models for Trust Management System

The major objective of our trust management system is to detect as many malicious and selfish nodes as possible by calculating their trust values in order to prevent them from influencing the network. Based on the trust values, the nodes can be simply labeled as trusted or untrusted. The former are considered to be the ones that normally complete the services requested of them such as message delivery, file sharing, etc. The latter are either the malicious ones that attempt to provide abnormal services to attack the network or the selfish nodes that solely focus on saving their own profits by partially providing services or not providing them at all.

To accurately discriminate between trusted and untrusted nodes, one needs to calculate the trust values precise enough to portray the nodes' trust status. Therefore, an exact calculation of the trust values is important in TM modeling.

In this chapter, the ideas on how AR and ARX models work in analyzing TM systems will be explained in detail.

4.1 Assumptions and Notations

4.1.1 Assumptions

To cast the proposed models into a trust management framework, there are some assumptions that need to be predefined:

- The assumption on nodes: We assume that there is a large number of nodes, which are randomly deployed in a wireless ad hoc network. Each node has a unique identity given by its unique location, so that every peer can be properly identified during any pairwise communication and its trust values can be properly evaluated by one other. There are no energy limitations for every peer to complete its message transmissions.

- The assumption on systems: In this network, the service peers mainly provide is message transmission, and we suppose that every peer can overhear the position of its neighbor's nodes during transmission, forward the message or drop it. In this way, each peer can generate an observed trust value about the neighbors with which it has interacted after the message delivery. We assume that the message transmission path is built through Greedy routing [16]. This protocol aims at finding the neighboring node with the minimum distance to the destination node.

- The assumption on message dropping by nodes: A node is allowed to drop a message only when its behavior is not trustworthy; in this way, we can focus on the influence of the node's trust degree on the model's performance.

4.1.2 Notations

Consider a wireless ad hoc network that contains nodes $N = \{1, 2, ..., n\}$. Every node maintains a list containing its one-hop neighbors and a time-varying set of trust values per neighbor.

$T(t)_{i,j}$: The observed trust value of node i towards node j based on their communication at time t.

$NeighborNode(i)$: The set of one-hop neighbors of node i. It helps node i select the neighbor with shortest distance to the destination node, since we use greedy routing to build a message transmission path.

$RP(t)_{i,j}$: The reputation value of node i towards node j at time t. It is used to predict node j's future trust value.

4.2 Trust Value and Reputation Value

We place our proposed models, i.e. AR and ARX, into the realm of reputation-based TM systems so as to analyze the confidence problem in a wireless ad hoc network. A reputation-based system calculates the reputation value of the nodes by gathering observations about their behaviors over a period of time. When a peer wants to establish a trust relationship with neighbors, it always predicts the trust value of its neighbor to evaluate how trustworthily it will behave in future interactions. The trust value is often computed by reputation over a period of time, i.e. *Trust value*=$\Phi\{$*Reputation, the Time elapsed since the reputation was last modified*$\}$ [4], where $\Phi\{\}$ is the trust function used to calculate the trust values. Here, Φ is the function in AR or ARX model.

4.3 AR Model for Trust Management

In AR model, the predicted trust value is only based on direct observation, which is generated by a node towards the neighbor being assessed on the basis of their past communication history.

4.3.1 Basic Model Design

At first, the nodes are unfamiliar with each other and the trust value for each node is '1', which indicates total trust. At the beginning, every node can communicate with one another. There is a source node i that wants to send a message to a destination node j through several intermediate nodes. The source node first finds its forwarding neighbor by looking at the $NeighborNode(i)$ list and calculating the distance from each neighboring node to the destination node. Then, it chooses the neighbor with the shortest distance to the destination node and proceeds to evaluate its trust value based on its mutual history of cooperations. If the predicted trust value is high enough, which means that this node will perform well with a high probability, then it should be chosen as the next intermediate node to forward the message. Otherwise, the source node should exclude this node and go on to find the next neighbor node that has the minimal distance and proceed to its evaluation. The steps above should be repeated. In this case, we could find a node that has both short distance to the destination node and high trust value to guarantee that the message would be successfully transmitted.

Cooperation can take place with the selected node in terms of message forwarding. After their communication, the nodes will evaluate each other on how well they have done their job. If a node has not performed satisfactorily at time t, it will be assigned a low trust value, which will be updated in evaluator nodes' lists at time t. In this way, after numerous communications from time to time, nodes will build their trust value lists. With growing knowledge of the other nodes' behaviors, one node can establish the reputation value for other peers. It is defined as the average value of the direct observations taken during a period of time and denoted by $RP(t)_{i,j} = \frac{\sum T(t)_{i,j}}{T}$, where T is the time slot duration.

If node i wants to communicate with node j, i will use its reputation value for j, which is based on their past cooperations and predict the behavior that j will exhibit in future communication endeavors by calculating the trust value based on

these reputations. AR calculation in the AR model can be used to predict this behavior in the form of a trust value. Based on the descriptions of AR models in Section 3.1, the AR model for the TM system could be expressed as:

$$\hat{T}(t)_{i,j} = E(t) + \sum_{p=1}^{na} \varphi_i RP(t-p)_{i,j} \tag{4.1}$$

where φ_i is the coefficient of the model, na is the order, $E(t)$ is white noise disturbance value and $\hat{T}(t)_{i,j}$ is the predicted trust value of N_j from N_i. For the introduction in Section 3.3, we could use LS to estimate the coefficients based on the given reputation values.

There exists a common misconception about AR order, i.e. that with larger order the prediction of an AR model is more accurate, since a larger order means more past reference data. However, real observations of the AR model order disproves this belief. From Figure 4.1, we can tell that increasing the order from the 9th value onwards causes the drop rate (percentage % of packets dropped) to grow. This is because past reputation values are out of date and thus not suitable for predicting current trust values. The old reputation values can interfere with the accuracy of the prediction. Therefore, AIC is indeed a good mechanism to determine the order of the model. It aims to find the best model with the smallest order.

4.3.2 Evaluation

Although we can get a predicted value by performing the computations above, without evaluating the formula, we do not know when to choose the node to begin the communication. In this section, we will use a thresholding technique to evaluate the trust value.

We define a threshold δ that every node should know. When the predicted value $\hat{T}(t)_{i,j}$ is above δ, we will let node i view the node j as trustable and start a communica-

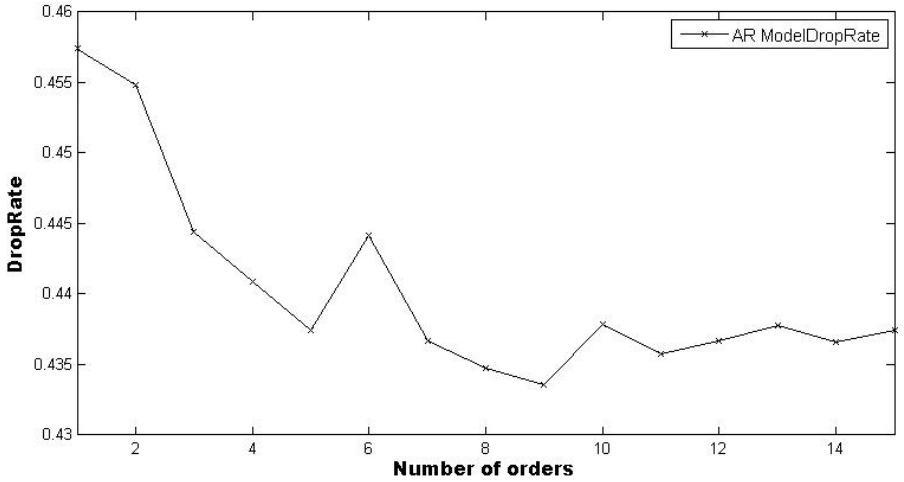

Figure 4.1: Drop rate with different order.

tion with it. After that, i and j can update their trust value lists with the observation values. If the predicted value is below the threshold, i will abandon j and go on to find another node providing the same service. The threshold directly determines whether a node can be selected as a cooperator. So it is important to define an appropriate threshold value. If it is too small, then a malicious node may be chosen to engage in a cooperation and it can damage the network. If it is too large, then a lot of nodes fall in increasing the chance of being active in the network, but only a few nodes with high trust values will be maintaining it. In our model, we will define the threshold δ as $\overline{T}(t)_{i,j}$. Since one $\overline{T}(t)_{i,j}$ is only valid for a single na time slot, we need to change δ as the slot changes and, consequently, the trust value list of every node must be updated at the end of every time slot.

This scheme determines the threshold value better than predefining it, for it can change later on depending on the observation values that fit the dynamic system. It is more flexible than a fixed threshold method and can select the nodes in a more precise way.

4.4 ARX Model for Trust Management

AR models only focus on the direct observations of a node. They are subjective and insufficient. Using the AR model also causes a loss in the supervision function of a trust management system because a node will never know another node's past behaviors until their communication is made personal. If a node is a good one, then the communication will be beneficial. Yet if it is a misbehaving node, this communication may cause potential damage to the system. To prevent this from happening, the ARX model uses an input clause to establish a supervision function upon the system in the form of indirect recommendations provided by peer nodes that also have communication with the node under assessment. Therefore, ARX models calculate the trust values based on both the direct observations and indirect recommendations from other nodes.

4.4.1 Basic Model Design

At the outset, the trust value of every node is '1', which means total trust as in the AR model. However, with the recommendation part, a supervision function is enhanced. For example, if node i did not communicate with node j which was previously malicious, in AR model node i would take node j to cooperate, which could lead to a message drop or a system attack. However using the ARX model, there may exist a couple of neighbors of node i that might have communicated with node j before and so they could give a recommendation to node i saying that j is untrusted by giving a specific low trust value. Based on the description in Section 3.2, the ARX model for TM systems can be expressed as:

$$\hat{T}(t)_{i,j} = E(t) + \sum_{p=1}^{na} \varphi_i RP(t-p)_{i,j} + \sum_{p=1}^{nb} \mu_i R(t-p)_{i,j} \tag{4.2}$$

where φ_i and μ_i are the coefficients of the model, na is the order of the outputs, nb is the order of the inputs, and $R(t-p)_{i,j}$ is the combined recommendation value from

node i's neighbors for node j at time $t-p$, which will be discussed in Section 4.4.2. Also, the recommendation value is time-varying. Every recommendation is offered in correspondence with the reputation value at a given time. Therefore, we can provide a time series of both reputation and recommendation values for trust value prediction.

Before estimating the parameters, we should determine the orders (na,nb) of the model. An appropriate order selection is crucial to the ARX model. If the orders or the historical data are low, then our trust value prediction will not be precise because of a lack of references. This, in turn, may cause a node to choose a liar peer. Otherwise, if the number of the orders are too large, it means that the nodes need to get a lot of information through estimation and that may bring about node overload. In our ARX model, we will choose AIC (Akaike information criterion) [2] to help determine the orders.

4.4.2 Recommendation Value from Other Nodes

When node i wants to estimate how well node j will provide a honest service, its own direct experience is sometimes subjective and not enough, so we need objective opinions to make the estimation more precise. Recommendation can help to solve this problem.

In the analysis above, each node has a list of observation values for the nodes it has communicated with in the past. When node i wants to assess node j, then the observation value i maintains on j is based on direct observed experience. The other nodes, for example, $k1$, $k2$, ..., kn are the neighbors of i and have observation values about j based on their cooperations with j. These values are the recommendations from $k1$, $k2$, ..., kn, denoted as $RP(t)_{k1,j}$, $RP(t)_{k2,j}$, ..., $RP(t)_{kn,j}$ respectively, as shown in Figure 4.2.

However, this scheme still cannot guarantee a precise prediction if it is rooted solely on the recommendation values without any further reference. Not every node's

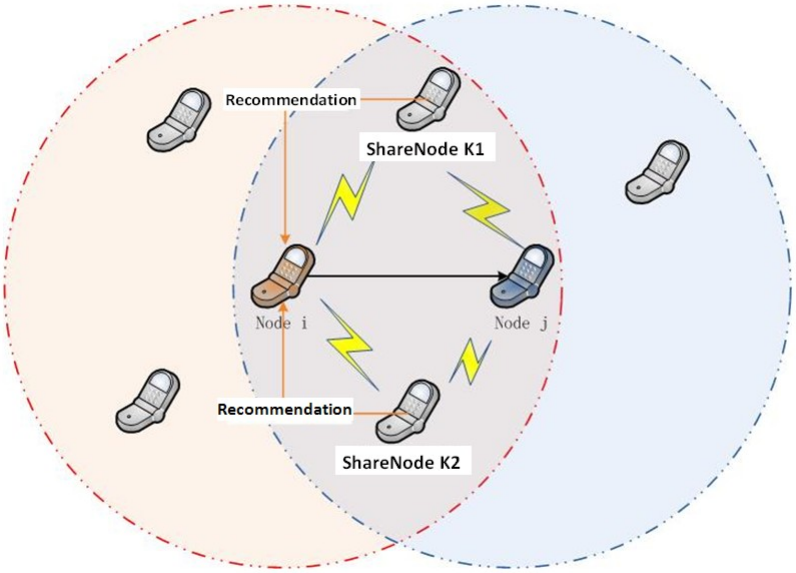

Figure 4.2: Recommendation selection

recommendations are trustworthy. Some nodes may provide false recommendations for malicious purposes, especially in resource competition systems. For example, in a credit system, resource like messages could help a node gain credits. There exists intense competition for resources among nodes. If a node wants to hold more exclusive resources and does not want other nodes to get them, it is more likely that it will provide false recommendations with the intention of disrupting the communication flow between the resource provider nodes and the nodes under critical assessment. This would help ensure that the latter cannot get resources.

In order to prevent these kinds of node influences upon the trust value prediction process, recommendations should be assessed as well. Nodes that lie about recommendations should be assigned low values so that they cannot influence the estimation too much. Furthermore, malicious and selfish nodes with low reputations should not affect the estimation either. Nodes in either situation may not have a high reputation after several cooperations with their neighbors. So we can weight the recommendations by each node's own reputation values. For example, if $k1$ wants to provide a recommendation $RP(t)_{k1,j}$ to i about j, it should be weighted by i with the reputation value

$RP(t)_{i,k1}$ from i towards $k1$, represented as $RP(t)_{i,k1} \times RP(t)_{k1,j}$. In this case, the misbehaving nodes cannot be active in trust value prediction.

Considering this issue, we define the recommendations offered to node i at a certain time t as:

$$R(t)_{i,j} = \frac{RP(t)_{i,k1}RP(t)_{k1,j} + RP(t)_{i,k2}RP(t)_{k2,j} + ... + RP(t)_{i,kn}RP(t)_{kn,j}}{RP(t)_{i,k1} + RP(t)_{i,k2} + ... + RP(t)_{i,kn}} \quad (4.3)$$

With the reputation values that node i has about nodes $k1$, ..., kn who give the recommendations, we can assign high weights to the trustworthy nodes so as to let them exercise more influence in the calculations and, analogously, grant low weights to the non-trustworthy nodes in order to minimize their effect. $R(t)_{i,j}$ is the final recommendation value we use in the model at time t.

4.4.3 Model Adjustment

The model, namely the parameters, cannot remain constant over time. The parameters can only offer a precise model with reputation values within a particular time frame. When more recent reputation values are updated, the parameters for the last period of time cannot fit the new slot of reputation values. This means that the model cannot accurately describe the system any more. If we continue using the same parameters for prediction, the difference between measured observation values and predicted trust values will turn larger as time passes by. Therefore, we need to calibrate the model after certain time intervals. If the result meets the minimum validation requirement, e.g. a specific data threshold, then the model is considered to work well. Otherwise, the model needs to be adjusted by using the recent updated reputation values and recommendation values.

To validate the model, we employ the coefficient of determination or R^2 presented

in [30]. According to [30], R^2 is used to measure how well a model could predict a future performance based on comparing the predictions it has made with the corresponding tangible outcomes. In a linear regression, R^2 could be simply defined as the square of the differences between predictions and outcomes, represented as Equation 4.4:

$$R^2 = 1 - \frac{SS_{err}}{SS_{tot}} \tag{4.4}$$

where SS_{err} is the sum of squares of residuals, i.e. the difference between outcomes and predictions and SS_{tot} is the total sum of squares of differences between outcomes and the average of the outcomes.

With the corresponding values of our model, we can change the representation of R^2 into:

$$R^2 = 1 - \frac{\sum_{t=1}^{na} \varepsilon(t)_{i,j}^2}{\sum_{t=1}^{na} (T(t)_{i,j} - \overline{T}(t)_{i,j})^2} \tag{4.5}$$

where

$$\varepsilon(t)_{i,j} = T(t)_{i,j} - \hat{T}(t)_{i,j} \tag{4.6}$$

$\overline{T}(t)_{i,j}$ is the mean of the measured trust value in na time slot.

The R^2 indicator expresses how close the measured value is to the predicted value.

Chapter 5

Simulations and Performance Evaluation

This chapter covers the details of the simulation of the proposed models on *MATLAB/Simulink* platform using the *MATLAB* System Identification Toolbox and sheds light on their performance evaluation in Section 5.3.

5.1 Using MATLAB System Identification Toolbox

System Identification Toolbox is used to construct and estimate the mathematical models for dynamic systems with the measured input and output data from the systems. System Identification Toolbox contains model structures of low-order process models, transfer functions, state-space models, linear models with static nonlinearities at the inputs or outputs, and linear and nonlinear autoregressive models.

With this toolbox, we can analyze and process the measured data, determine the model's structure and order, then estimate the model's parameters with the measured data and finally validate its accuracy. After importing the input and output information into the system, we could make use of the toolbox functionality to process the data and estimate the model's parameters, e.g. for autoregressive models (ARX, AR).

When using data processing functions in our ARX model, the order is set to be a vector of three integers. The default order is $[na, nb, nk]$, where na represents the number of historical direct reputations required in our simulations, nb is the number of historical recommendations from other neighbors and nk the number of recommendation samples loaded in the system, after which the recommendations begin to affect the prediction. We set nk to its default value 1 so as to let the recommendations influence the output prediction right from the outset of the simulations.

5.2 Simulation Setup

There were 100 nodes randomly deployed into a rectangular area of 100×100 m^2, as shown in Figure 5.1. The communication radius of every node is 10 m. It is assumed that each node is capable of bidirectional cooperation and has enough energy to fulfill every interaction.

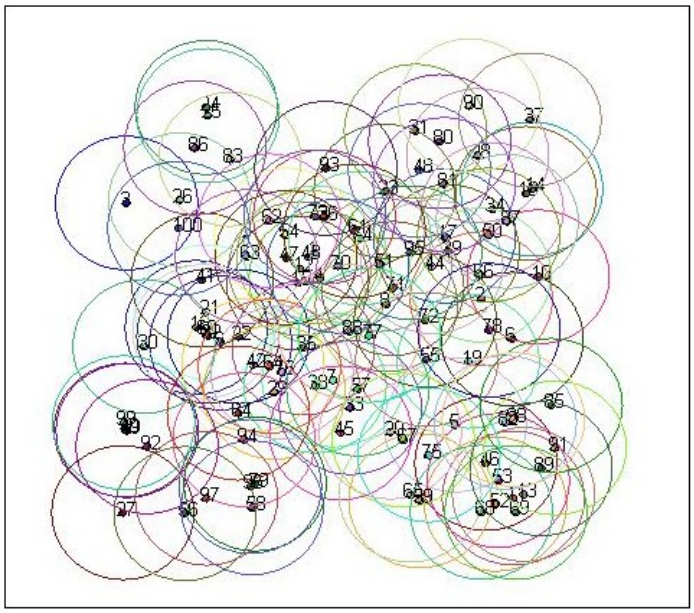

Figure 5.1: Network Topology

A total of 600 data communication sessions starting at arbitrary times were simulated. Each session occurs between a randomly selected pair of nodes and involves 100

data packet transmissions at a constant rate of one packet per simulated time unit. We randomly select a certain number of nodes (except source and destination) as malicious nodes. They drop every received packet with 70% probability. Normal nodes drop packets with a small probability of 15%, thus simulating an unreliable wireless communication medium. Each (trust observation) time slot is composed of 5 consecutive simulated time units; in other words, each data communication session spans across 20 time slots. The model validation is carried out every 5 time slots and the model update is conducted immediately if the validation result indicates so.

5.3 Simulation Results

In this section we evaluate the performance of the ARX, AR and BRSN models by executing them in wireless network simulations along with a Greedy [16] routing scheme. Firstly, the network density is altered in Section 5.3.1 to gauge how it affects each model. Secondly, the number of malicious nodes is varied in Section 5.3.2 with the goal of corroborating how stable and efficient each model is. Thirdly, in Section 5.3.3, the importance of setting appropriate model parameters is discussed by evaluating the models' performance under different orders and update periods. Fourthly, according to the previous discussion, the optimal model settings for the network are derived. Then under these best settings in Section 5.3.4, the performance of the network with or without trust models is observed. This allows to check how trust models help the network reduce the message drop rate (percentage % of packets dropped). Finally, in Section 5.3.5, the overhead of these three trust models in dense and sparse networks respectively is contrasted. By focusing on the message drop rate, the prediction values and the prediction errors, an insightful comparison among the ARX, AR and BRSN models can be carried out. Furthermore, by referring to values reported by AIC and FPE, the best model can be selected.

5.3.1 Effect of Network Density on Message Drop Rate

In this section, we will focus on the message drop rate of the network in order to evaluate the performance of each trust model. As demonstrated in Section 5.2, the message drop rate of each trust model is calculated during 600 data sessions. To modify the network density, the total number of nodes in the fixed area is varied. In this way, the number of neighbors that provide recommendations varies too. This allows shedding light on how the recommendations affect the overall performance of the trust models.

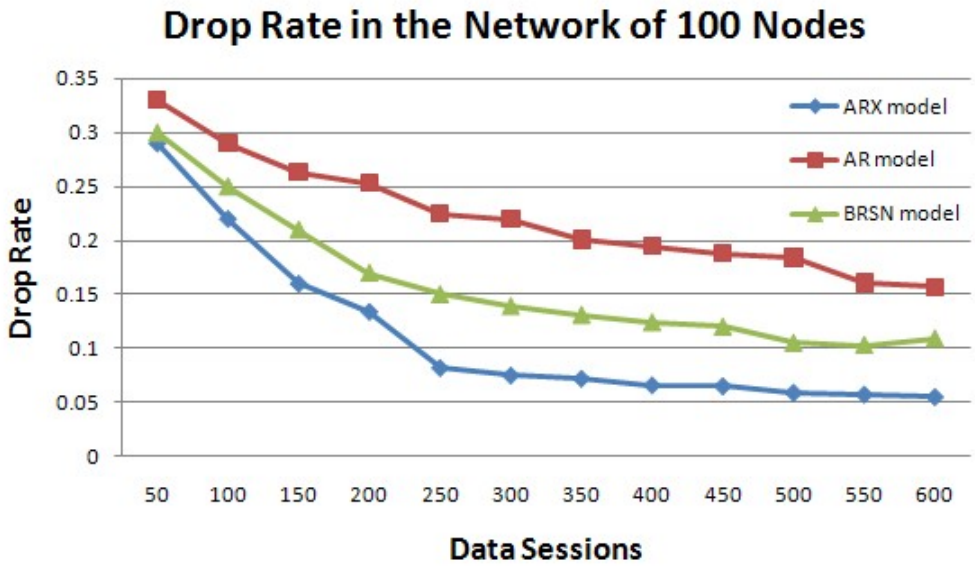

Figure 5.2: Drop Rate in the Network of 100 Nodes

With a varying number of recommendations, the ARX and BRSN models would exhibit different performances. However, AR's behavior will not change too much because it does not need any recommendations. In order to evaluate the impact of recommendations over the models' performance, the number of nodes is set to 100, 90 and 80 with a constant 15% malicious node rate. By doing so, we vary the density from dense to sparse and hence reduce the number of recommendations. The remaining parameters are fixed as follows: the ARX model's order is set to the best fit value $[4, 3, 1]$ and update period equals 5 time slots; the AR model's order is set to the best fit value

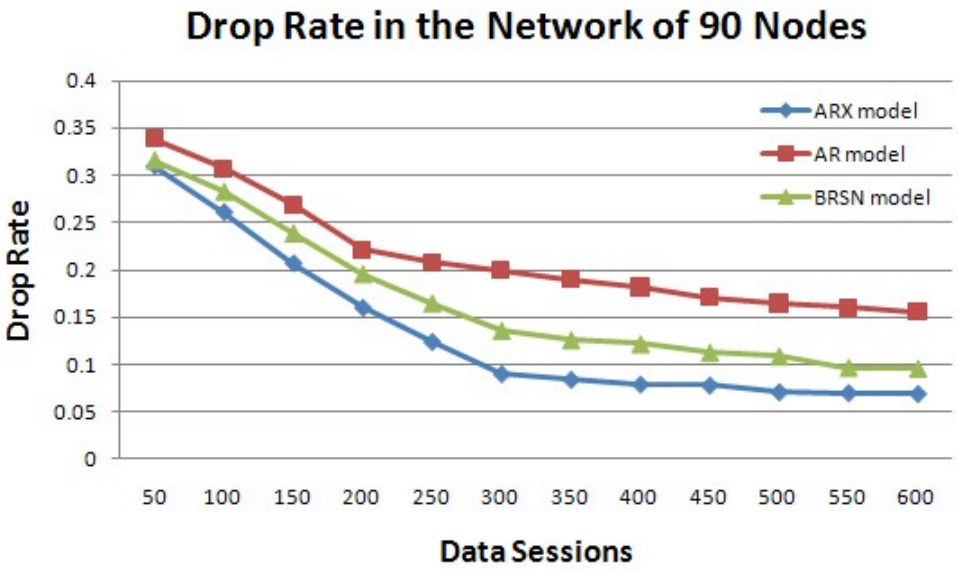

Figure 5.3: Drop Rate in the Network of 90 Nodes

6 and update period to 5 time slots.

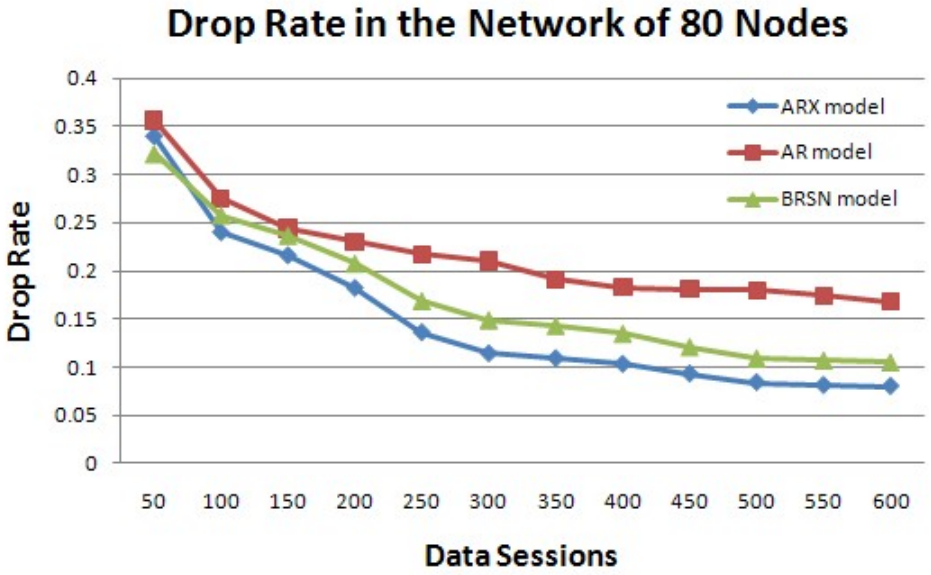

Figure 5.4: Drop Rate in the Network of 80 Nodes

Figure 5.2 to Figure 5.4 display the message drop rate with varying network density. In a dense network, a larger number of recommendations make ARX and BRSN models achieve a lower drop rate than AR. Moreover, ARX reaches the stable state

faster. For example, ARX maintains a stable drop rate after 250 data sessions when 100 nodes exist in the network. In a sparse network, a lower number of recommendations make ARX and BRSN models attain a similar drop rate to AR. For example, the drop rate of these three models are very close to each other at the early stage when there are only 80 nodes in the network. Notice that AR's drop rate is almost the same in these three figures as expected. Meanwhile, even with a small number of recommendations, ARX is always the best trust model among all under consideration. ARX was also able to help build reliable relationships among nodes more quickly with more recommendations.

5.3.2 Effect of Malicious Nodes on Drop Rate

To validate how reliable and stable the trust models behave, the number of malicious nodes in the network is modified in the range 5% to 20%. The number of nodes is fixed to 100 so that the number of recommendations will remain unchanged. Other parameters are set as follows: the ARX model's order is set to the best fit value $[4, 3, 1]$ and update period to 5 time slots; the AR model's order is set to the best fit value 6 and update period to 5 time slots.

Figure 5.5 to Figure 5.7 show the message drop rate in the network with 5%, 10% and 20% of malicious nodes respectively.

Comparing these three figures, one can realize that all three trust models managed to control the drop rate at the lowest level when there are only 5% malicious nodes. As this percentage goes up, the drop rate is also growing from Figure 5.5 to Figure 5.7. Moreover, the time for reaching the stable state is getting longer because the trust models will require more time to identify a larger number of malicious nodes. One may also conclude from the three figures that only ARX can still maintain the message drop rate around 5% regardless of how many malicious nodes are there. With more of these nodes in the network, however, the drop rate of BRSN increases from around 10% to

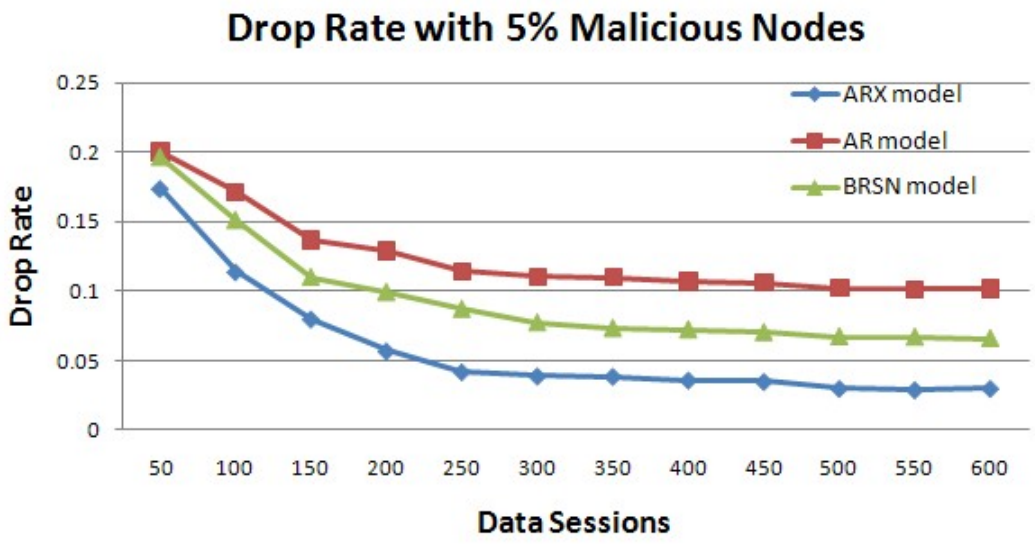

Figure 5.5: Drop Rate with 5 % Malicious Nodes

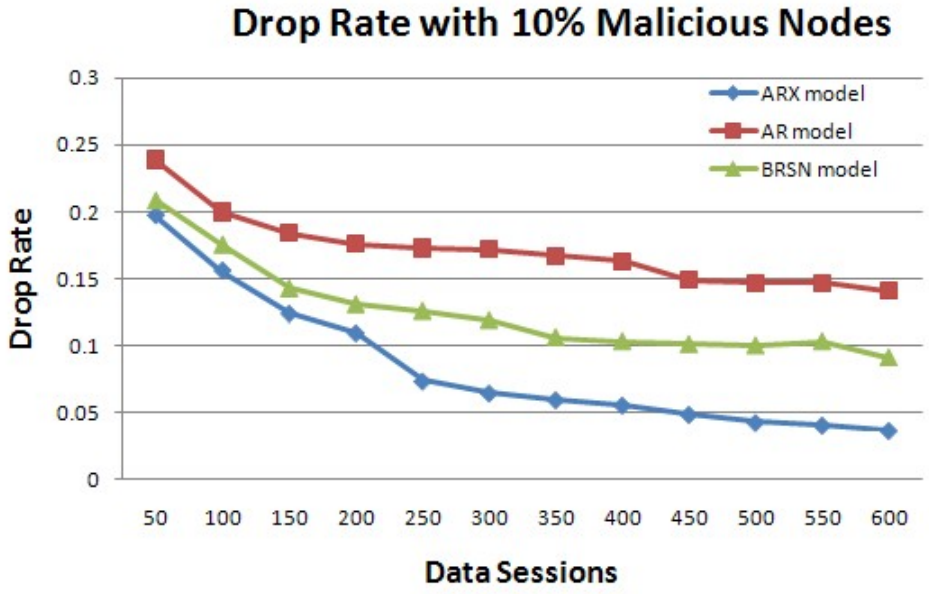

Figure 5.6: Drop Rate with 10 % Malicious Nodes

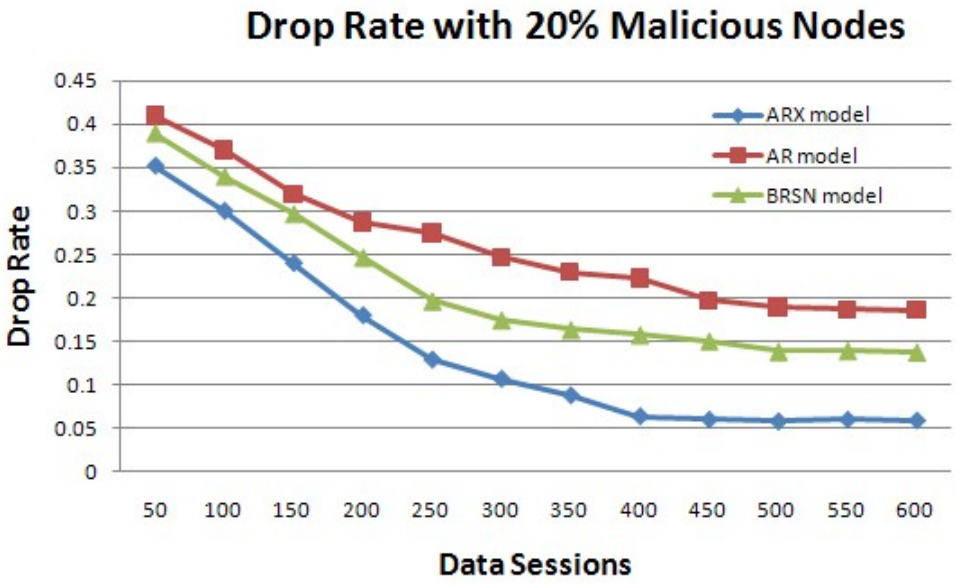

Figure 5.7: Drop Rate with 20 % Malicious Nodes

15% and that of AR also surges from 10% to 20%. Therefore, we could deduce that the ARX model is always able to work steadily even when a larger number of malicious nodes permeates the network.

5.3.3 Effect of Model Parameters and Model Accuracy

In this section, we will talk about the influence of the model' parameters on the network performance by evaluating the prediction errors. We will also analyze the model's property by referring to the AIC and FPE data. The prediction error is useful to measure the difference between the predicted trust values and the observation trust values. If we get a small prediction error value, it means that the predicted trust value is close to the observation trust values, then we could tell that the model's prediction is so accurate that the system could identify the malicious nodes to avoid the attack. The model's parameters mainly consist of the model's order and update period, which are not present in the BRSN model.

In Sections 5.3.3.1 and 5.3.3.2, the prediction errors under different parametric settings are contrasted. The best setting that leads to the lowest prediction error is

sought as well. Our result is also verified by using the best set when comparing the predicted values versus the observation values in Section 5.3.3.3. Then, with the best setting at hand, the model's property in Section 5.3.3.4 is discussed.

5.3.3.1 Effect of the AR Model Parameters on Prediction Error

The order of the model must be carefully selected. A large order value causes the prediction to mostly rely on the previous reputations, which may lead to a prediction lag. On the other hand, a small order value cannot provide enough information for a precise prediction. Figure 5.8 unveils that the prediction error decreases before AR model's order reaches 6. After this point, the prediction error becomes visible. It attains its lowest value when the AR model's order is 6 while the network performance peaks up.

Besides the model's order, the update period is important as well. It determines how often we change the model coefficient to fit the new reputation values. Figure 5.9 discloses the relationship between the prediction error and the model's update period. The curve reaches the lowest point when the update period is 5. Therefore, if we update the model every 5 time slots, we could accomplish the smallest prediction error. If we update it too often (i.e. update period is too small), we calculate the coefficient using only a little piece of information, which may bring forth an inaccurate prediction. Furthermore, updating the model too often costs the system a large amount of computational effort and storage memory. On the contrary, if we set a long update period, the model will quickly turn out-to-date given that it will not fit the newly updated reputation values, thus leading to a decline in prediction accuracy.

5.3.3.2 Effect of ARX Model Parameters on Prediction Error

The order of the ARX model consists of three parts: $[na, nb, nk]$, where na is the order of the direct reputation, nb is the order of the recommendation and nk is the number

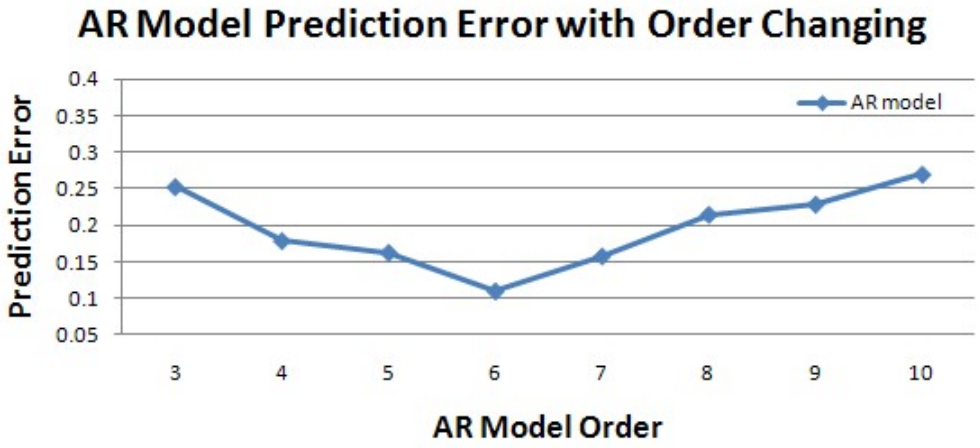

Figure 5.8: The Prediction Error with Changing AR Model Order

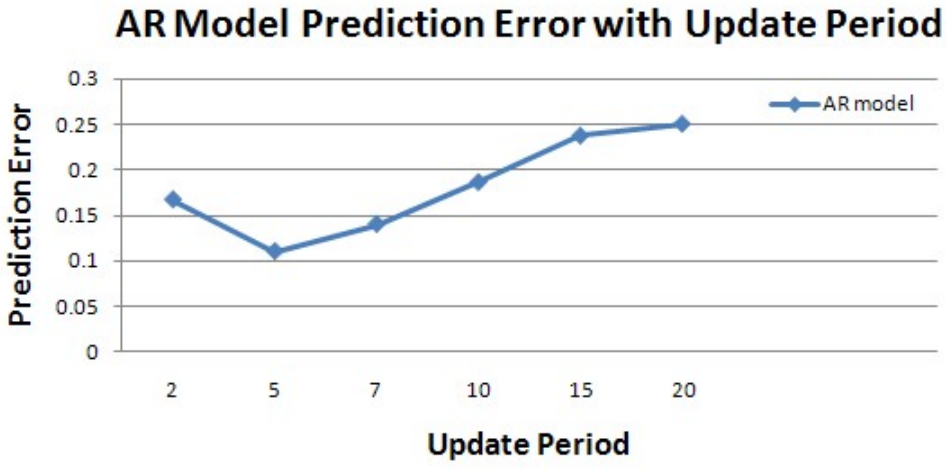

Figure 5.9: The Prediction Error with Changing AR Model Update Period

of recommendations that occur before they actually affect the prediction, which is set to the default value 1. So we vary na and nb from 2 to 10 to behold the change in prediction error. Figure 5.10 reports that when the order is $[4, 3, 1]$, the prediction error is the smallest. This order indicates that the prediction is the most accurate when it relies on the proper number of objective recommendations instead of on the subjective direct reputations. If we increase na, the prediction becomes inaccurate because the information is too subjective. Otherwise, if we increase nb, the prediction is not precise because it leans too much on the recommendations from other nodes, which may not be trustworthy.

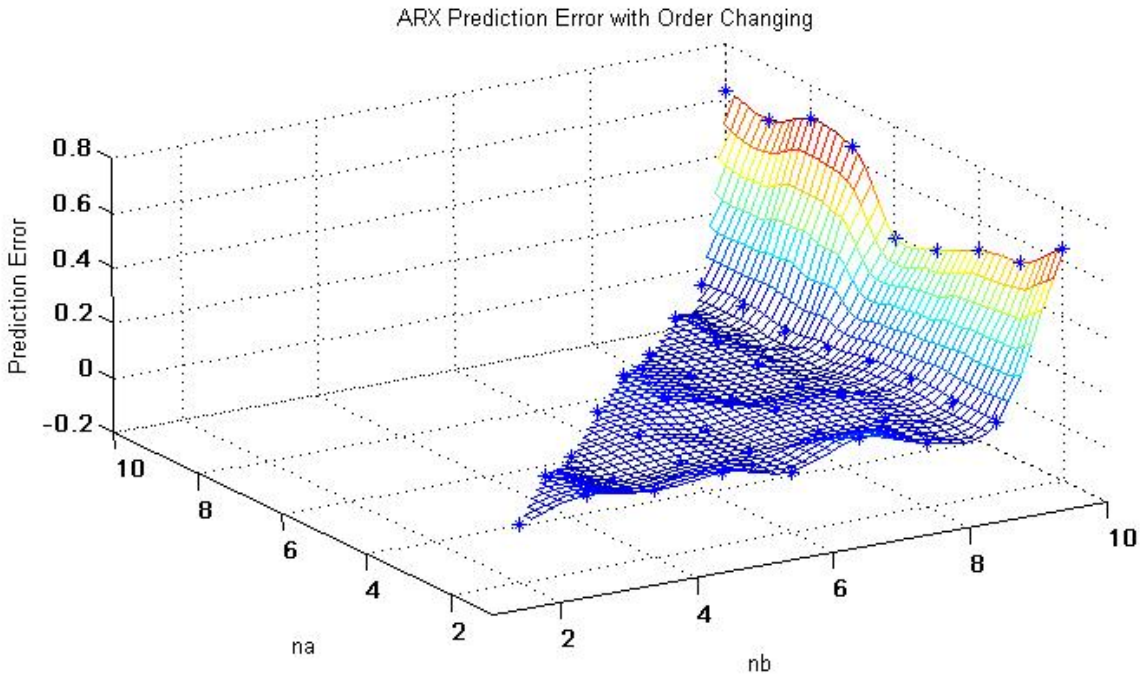

Figure 5.10: The Prediction Error with Changing ARX Model Order

Similar to the ARX model, the best update period for AR is 5 time slots as Figure 5.9 indicates. This result is in perfect alignment with our experimental setting in Section 5.2, in which the model should be validated every 5 time slots.

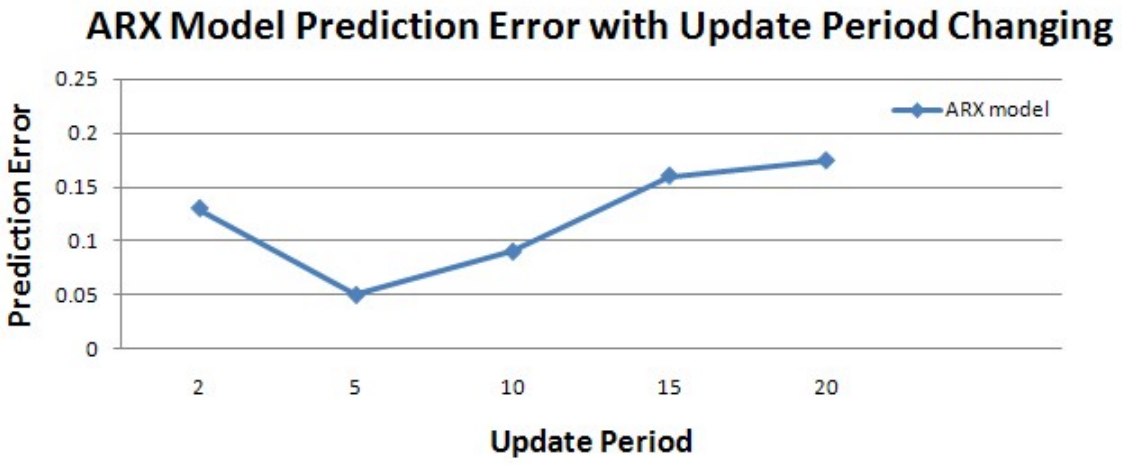

Figure 5.11: The Prediction Error with Changing ARX Model Update Period

5.3.3.3 Model Parameters Evaluation on Prediction Values

In Sections 5.3.3.1 and 5.3.3.2, the best parameter setting for the AR and ARX models were identified. So in this Section, these settings will be verified by considering the prediction values. Because the AR model is similar to ARX model in data processing, we only focus on ARX's prediction values.

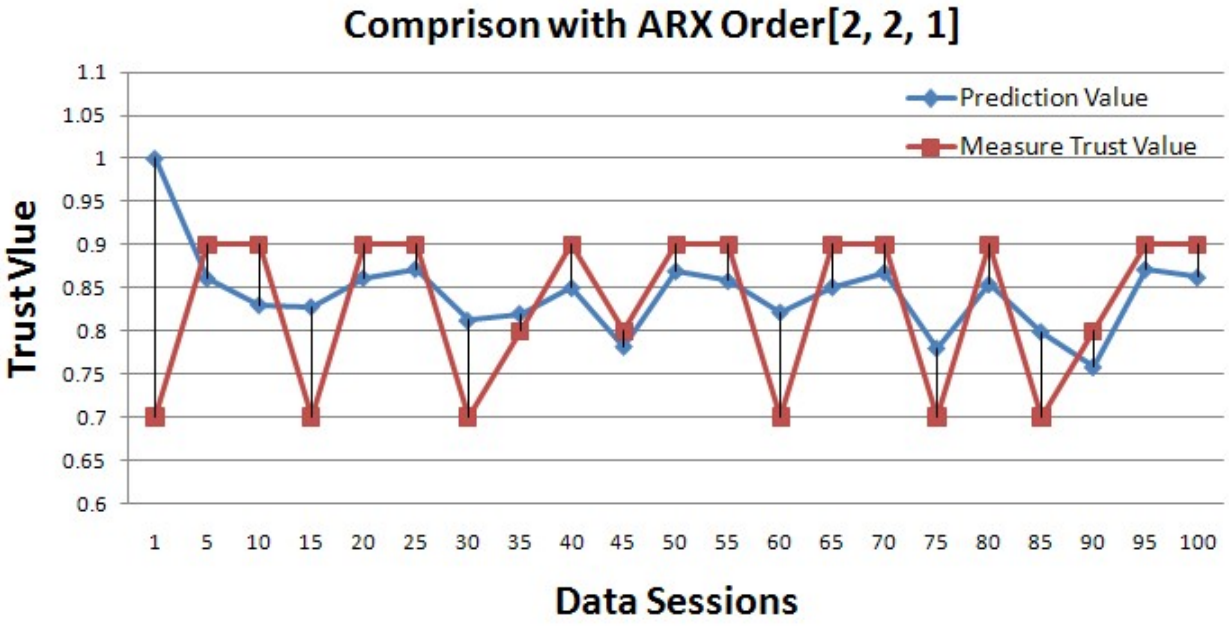

Figure 5.12: The Prediction Values with ARX Model Order Set to [2, 2, 1]

In Section 5.3.3.2, it was assumed that the prediction does not have enough information to be accurate if it involves few direct reputations and recommendations, so the prediction error is high. Figure 5.12 portrays the prediction values with the order fixed to $[2, 2, 1]$. From the figure, one may become aware that the prediction values hardly get close to the measured ones, especially when there is a large change in the latter. This is because there is not enough information for the ARX model to perform a forecast of the next trusted behavior. But with too much information (i.e. very high order values), the prediction is also not accurate because the redundancy information is overwhelming, as Figure 5.13 reveals. In Figure 5.13, the model order is set to $[7, 4, 1]$ yet the outcome is still unsatisfactory. The prediction has a latency when the measured value changes to a large extent, due to the fact that the model could not swiftly respond to the change in the out-to-date information. As discussed in Section 5.3.3.2, the best order is $[4, 3, 1]$. From Figure 5.14, it is possible to verify that the prediction value and the measured value are close. When there is a big change in the measured values, the model could manage the prediction under a permissible error.

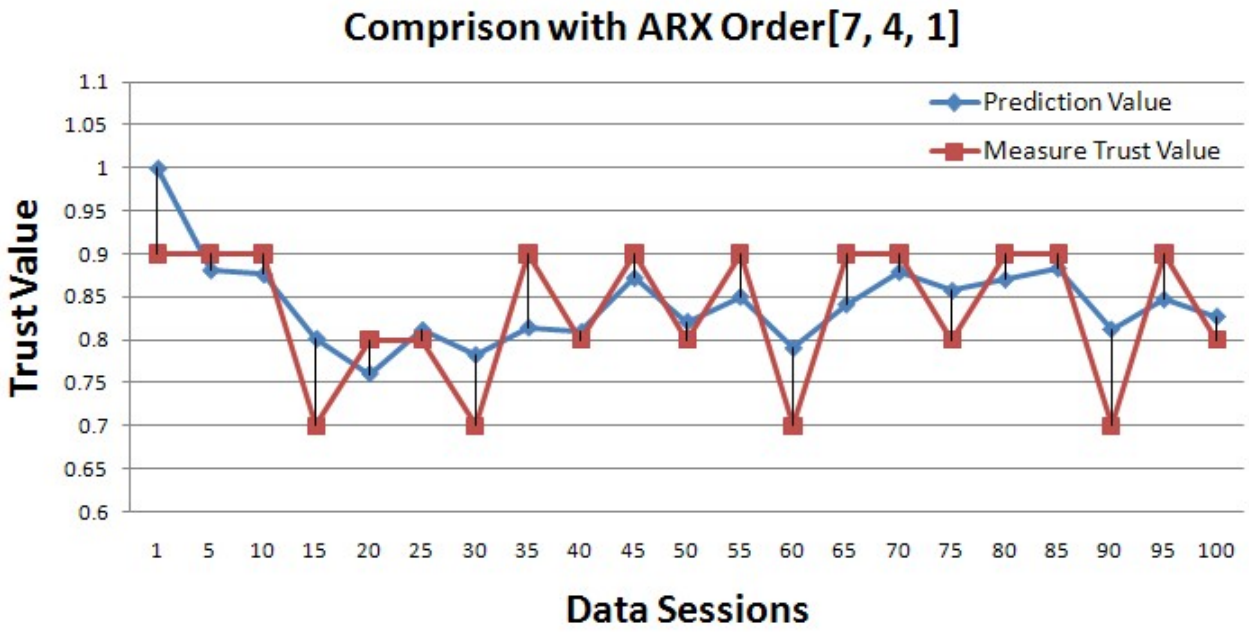

Figure 5.13: The Prediction Values with ARX Model Order Set to [7, 4, 1]

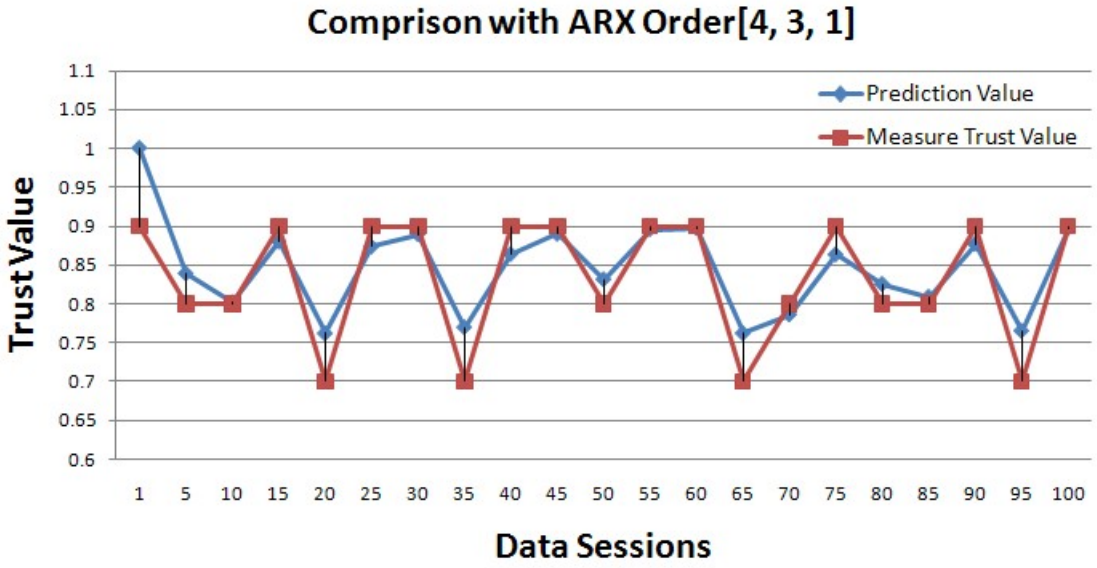

Figure 5.14: The Prediction Values with ARX Model Order Set to [4, 3, 1]

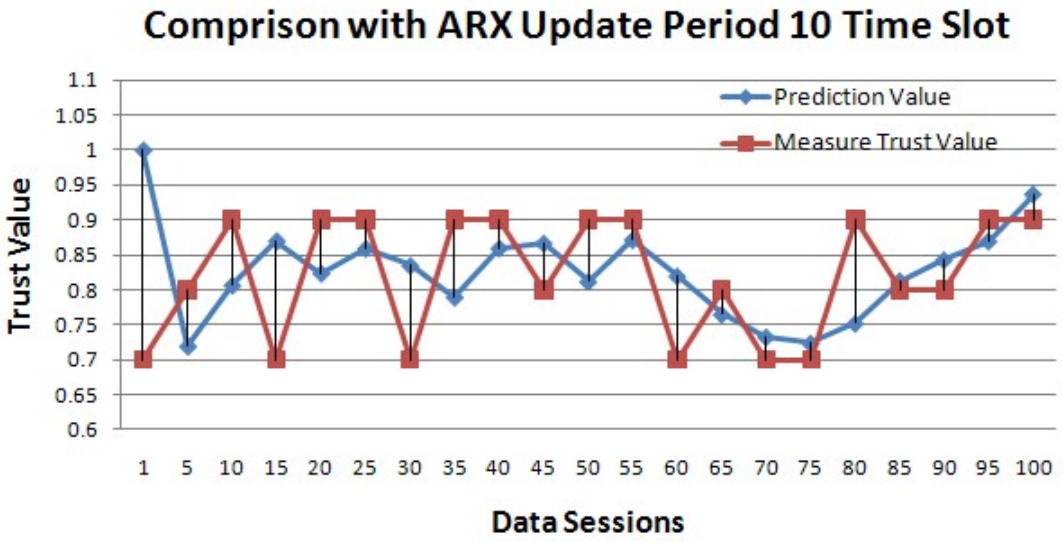

Figure 5.15: The Prediction Values with ARX Model Update Period Set to 10 Time Slots

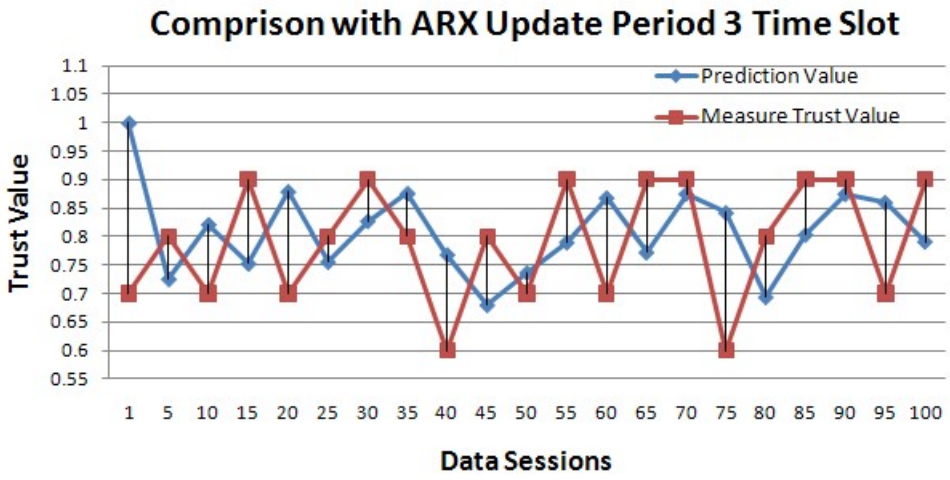

Figure 5.16: The Prediction Values with ARX Model Update Period Set to 3 Time Slots

As the update period changes, so it does the performance. Figure 5.15 compares the prediction values yielded with the update period equal to 10 time slots. In order to investigate the effect of the update period, the same order which is the best fit $[4, 3, 1]$ is kept. Compared with Figure 5.14 which exhibits the best update period 5 time slots, one could see that the prediction values in Figure 5.15 can not be confined to a stable range as in Figure 5.14. If the model is updated too often (with the update period set to 3 time slots as shown in Figure 5.16), the prediction value largely fluctuates when the measured value changes and also becomes more dissimilar from the measured value. This is ascribed to the fact that a long update period could render the model outdated so that it may not fit the new updated information. On the other hand, a short update period could render the information acquisition insufficient so that the prediction is easily influenced by the sharp change of measured trust values, as observed in Figure 5.16.

Therefore, by contrasting the prediction values with the measured values, it is possible to verify that the best fit order for the ARX model is $[4, 3, 1]$ and the best update period is 5 time slots. This finding fully agrees with the result in Section 5.3.3.2.

Table 5.1: Index for Comparison among the Models' Property

Model	AIC	FPE
AR model	-11.2660	0.0019
ARX model	-17.7852	0.0016
Bayesian	-11.5207	0.0077

5.3.3.4 Model Evaluation

In order to evaluate the model properties, each model was constructed under the best set, which is 100 nodes with 5% for each of them. The order and update period parameters are set to $[4, 3, 1]$ and 5 time slots for ARX model and 6 and 5 time slots for the AR model, respectively. Then the ACI and FPE indexes [2] were computed for the three models, as shown in Table 5.2.

The model with lowest AIC value best represents the time series (trust evolution) with a minimum number of parameters (i.e., minimum model order); the most accurate model has the smallest FPE value. As stated in Table 5.2, the AIC and FPE indexes of ARX are lower than those of the other two models, which means that the ARX model could best represent the trust evolution of a node with a tolerable number of output parameters with respect to the AR and BRSN models, therefore confirming our previous simulation analysis.

So, ARX renders a stronger representation than AR and BRSN models, which means that ARX could express the system's model more accurately so that it will lead to a better prediction performance.

5.3.4 Comparison under the Best Parameter Setting

In order to achieve a more solid work, 100 nodes were deployed in the system and randomly appointed 15% of them as malicious nodes so that there is a tolerable number

of attacks to the system. For the ARX model, the order is the best fit value $[4, 3, 1]$ and update period is set to 5 time slots. For the AR model, the order is also set to the best fit value 6 and update period to 5 time slots. Then the simulations in Section 5.2 were conducted. These three trust models built-in systems were compared with the non-trust-model-built-in system to judge to what extent is the trust model important for prediction accuracy and controlled drop rate.

5.3.4.1 System Performance Comparison

Figure 5.17 displays the message drop rate of Greedy routing (without any trust model), Greedy with BRSN [17], Greedy with AR and Greedy with ARX during 600 message rounds.

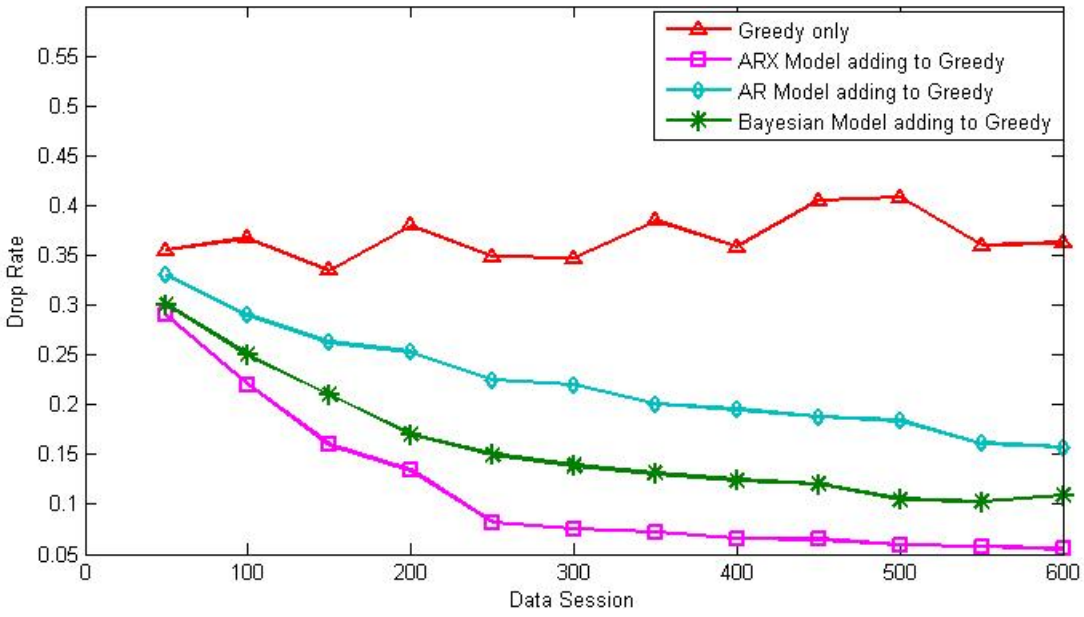

Figure 5.17: Message Drop Rate Comparison

Greedy combined with built-in trust models yields a lower drop rate than Greedy alone. Notice that the drop rates of different trust models are close to each other at the beginning but become distinct after several data sessions. This is because nodes can not predict the trust values with a lack of information at the beginning. After a

few interactions, however, nodes can foretell the trust values of others based on their past behaviors. Thus, the drop rates of built-in trust model systems fall. After around 250 data sessions, the drop rates stabilize.

The ARX model reaches the best performance as the drop rate goes down the fastest among all three models. BRSN leads to a lower drop rate than AR because it takes into account the second-hand information (recommendations) when computing nodal trust, which the AR model does not regard. Under the same condition, ARX does a better job than BRSN because its data source is anchored on a more direct numerical expression (weighting observations and recommendations) as opposed to the indirect expression in BRSN. ARX calculates the predictions on the basis of the reputations, whose values are the real observation trust values, so the direct data source expression could lead to a straightforward prediction computation. In BRSN, on the contrary, the prediction is derived out of two data sources, which represent the probability of good and bad behaviors. So the prediction only shows which behavior is more likely to occur instead of a trust value.

5.3.4.2 Prediction Values

In terms of analyzing which model can provide a better prediction, two out of 100 nodes were picked to calculate their observation trust values and prediction values over the last 70 interactions and pass them on to these three models. The number of malicious nodes was set to 15.

Figure 5.18 reports trust observations about a randomly selected malicious node in comparison with the trust estimates by all three models during the initial 70 data sessions. At the start, the initial opinions are all set to the default value 1, which means complete trust. Then the predicted values of the AR and ARX models are calculated from these initial values and become higher than the real observation values. This creates a big gap between the observation values and the predicted values from AR

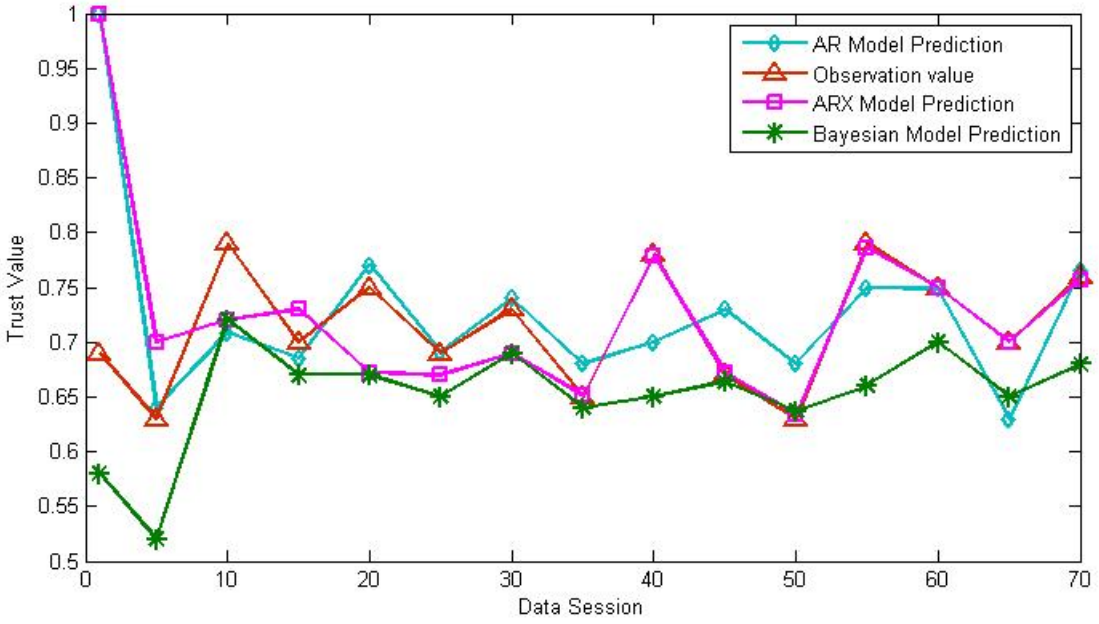

Figure 5.18: Comparison between Observation Trust Values and Model Prediction Values

and ARX. For BRSN, at the beginning, the predicted value is around 0.5, which means that both trusted and untrusted behaviors have the same probability to occur.

With a rise in the number of data sessions, the observation values also grow but not sufficiently (at first data session 30). In this phase, the estimates of all three models get close to the observations as expected, and AR is the fastest model to approach the observation trust values. BRSN performs similarly to ARX but less accurate than it.

After a certain number of communications (around data session 35) , the observation values are enough to compute more accurate prediction values. There is an obvious difference for the ARX model, as its trust estimates are very near to the observations, which indicates that it is the best model among all competitors. The AR model sometimes gives precise predictions, but the predicted values are not continuous over a period of time. This indicates that AR cannot work well for a long time. As for ARX, the observation values and the predicted values match each other perfectly, and last much longer than AR. One may conclude that ARX's predicted values are close to

the observation values and also can remain working well for a long time. Concerning BRSN, the predicted values seem to fluctuate because its update scheme only takes into account two parameters, which could lead to a large change from the previous value.

5.3.5 Model Overhead

The overhead of each model must be taken into consideration when assessing its performance. So in this section, we will focus on the memory and computation time associated with each model. We also discuss the overhead under both dense and sparse networks and its effect on each model.

5.3.5.1 Overhead Comparison in Dense Networks

To construct a dense network, we deployed 100 nodes and limit the number of malicious nodes to 5 % of the total number of nodes. The storage overhead contains all the variables that are pre-set and generated in the network. The variables' initializations are all the same for these three models, such as node location information. The calculation requires data types and steps of each model, which make the storage overhead different for each model. The time overhead is the extra time consumed to send a packet one hop distance in each model.

From Table 5.2, one may realize that the ARX model utilizes the largest storage space and also the longest precessing time. Yet this finding does not mean that more storage space costs more processing time. In fact, the time spent is related to the computational complexity of each trust model. For example, the AR model only uses $676KB$ space which is much smaller than the BRSN model, but AR model's precessing time is a little longer than BRSN's. It means that although the processed data in AR are fewer than those in BRSN, the former model entails more entangled computations than the latter and hence it is more time-consuming.

Table 5.2: Overhead Comparison in Dense Network

Model	Storage Overhead	Time Overhead
AR Model	676KB	7.25 ms
ARX Model	798KB	13.19 ms
BRSN Model	724KB	7.03 ms

Table 5.3: ARX model Node List

Variable Name	Description
Reputation(NeighborNodeNumber,100)	Observation trust values, the input of ARX model, could store 100 time slots values
Recommendation(NeighborNodeNumber,100)	Recommendation from share nodes of two neighbor nodes, could store 100 time slots values
Prediction (NeighborNodeNumber,100)	Model Prediction values.
Trust	Accumulator for calculating the Reputation every time slot.
Model	Store the current ARX model between two selected nodes.
NeighborNode	Store the neighbor node IDs

In the ARX model, the details for every node list are outlined in Table 5.3. It should store the reputation values, the recommendation values for the model computation, the prediction values for comparison with the observation values, the trust value as an accumulator for calculating the reputation value every 5 message units and the model's information for each node pair. In addition, AR model's list has almost the same variables that ARX model except that AR does not include the recommendation part.

For the BRSN model, Table 5.4 outlines its storage requirements. The list at each node is totally different from those in ARX and AR. It has two parameters (α and β) for model calculation, reputation and prediction parts for comparing results, and a trust value for updating the reputation. As the storage list displayed in the table, ARX model contains more information than the other two models, which results in

Table 5.4: BRSN model Node List

Variable Name	Description
Reputation(NeighborNodeNumber,100)	Observation trust values, the input of ARX model, could store 100 time slots values
Prediction (NeighborNodeNumber,100)	Model Prediction values.
trust	Accumulator for calculating the Reputation every time slot.
Alphaa(1,NeighborNodeNumber)	positive rating, it is updated every message transmission with r and s value.
Betaa(1,NeighborNodeNumber)	Negative rating, it is updated every message transmission with r and s value.
NeighborNode	Store the neighbor node IDs

bigger storage space. Considering the computation process, BRSN and ARX produce the prediction value based on both reputations (first-hand information) and recommendations (second-hand information), however AR only relies upon the reputation values which leads to a large reduction of the storage overhead.

When it comes to time overhead, ARX possesses the longest processing time among all models in this study whereas AR has the shortest processing time. If we examine the details about how each model works, we could notice that ARX bears the most complicated procedures on information requirement and data precessing. ARX and BRSN need both direct reputation of their own experience and the recommendation information coming from neighboring nodes. However, AR only needs its own direct information, so ARX and BRSN incur in a waiting time for the recommendation information from neighboring nodes and a processing time to merge the additional recommendation into calculation, as opposed to AR which does not embrace these two steps. Because the calculation and updating scheme for BRSN is much simpler than for AR, the former saves more time specially in this long execution of data sessions. ARX has the most embroiled calculation scheme and updates even more data than AR, which leads to an increase in the processing time.

Table 5.5: Overhead Comparison in Sparse Network

Model	Storage Overhead	Time Overhead
AR model	498KB	6.14 ms
ARX model	576KB	9.24 ms
Bayesian	524KB	4.77 ms

5.3.5.2 Overhead Comparison in Sparse Network

A sparse network is created by setting 80 nodes, with 5% acting as malicious nodes inside the network. Table 5.5 unfolds the overhead in this sparse network. Comparing with Table 5.2, one could tell that the memory and time are much lower than those in a dense network. Because a sparse network has less recommendations and information data, each model requires less storage overhead, especially for ARX, which exhibits a storage decrease over $200KB$. With less recommendation combinations in computation, the time for ARX and BRSN largely shrinks, but ARX is yet the most time-consuming model. There is an obvious time gap between AR and BRSN, which implies that the latter renders calculations faster than the other two models although with a superior storage overhead.

When comparing the drop rate in dense and sparse networks, the bottom line is that the recommendation information could help the ARX and BRSN models achieve a better performance but has nothing to do with the AR model. Additionally, by changing the parameters of the ARX and AR models, we observed the importance of the model settings: the high order and update period values would lead to a prediction latency whereas their low values would cause prediction inaccuracy because of the lack of information. The best parameters for each model are also derived by calculating the prediction errors and prediction values. Then with the best-fit parameters, we focus on the drop rate to evaluate the impact of each model on the system (cf. Section 5.17), and the prediction values to shed light on the accuracy of each model (cf. Section 5.18).

From the results, one may say that the ARX model yields the best ability to control the message drop rate and reach the stable state. Furthermore, it attains the most accurate prediction of the trust values, which are nearly identical to the measured values. However, the outstanding calculation ability of the ARX model comes at the expense of the largest storage and time overhead, while the AR model has the least overhead (cf. Section 5.3.5).

In light of the above facts, we choose our models depending on which aspect we want to consider. For example, if we are mainly interested in the model's accuracy, ARX model is the best choice; however, if we intend to reduce the system overhead, AR model would become our selection.

Chapter 6

Conclusions & Future Work

6.1 Conclusions

In this thesis, we proposed to use an autoregressive model (AR) and an autoregressive with exogenous inputs model (ARX) in wireless ad hoc networks for establishing trust relation between nodes on the basis of their prior interactions.

We analyzed the performance of the proposed AR and ARX models by changing the system density and model parameters, mainly observing the message drop rate and prediction values. Both AR and ARX models were compared with a known Bayesian non-autoregressive model named BRSN (Beta Reputation system for Sensor Networks). From the results, we concluded that AR model is efficient in establishing trust relations since it only counts on the direct interactions; however, it lacks supervision by other nodes. The ARX model with an addition of recommendation input part makes the supervision by other nodes available to build trust in a more reliable way but less efficient (requiring extra local space for storing the recommendations) compared to the AR model. Our simulations results show that the ARX model performs best in term of accuracy and improvement brought to Greedy routing. Although BRSN model has the

benefit of less information inputs and model complexity, it is not that efficient to build a reliable relationship between nodes like the autoregressive models without sufficient information about node historical behaviors.

6.2 Future Work

When a node has a low trust value, one below the threshold, it would never be taken for cooperation by other nodes even if it wants to provide a trustable service. Such nodes cannot participate in cooperation and only a few highly trustable nodes are available in the network to do useful work. When there is a need to send a large amount of data from one node to another, this will cause a problem. Therefore, we need to improve the ARX model by considering the frequency of nodes activity to give the node with low trust value more opportunity to communicate with other nodes and increase its trust value by providing trustworthy services. Also as future work, we plan to compare our proposed trust management system with existing trust management systems that use different approaches, other than Bayesian.

Bibliography

[1] H. Akaike, "Statistical predictor identification", Annals of Institute of Statistical Mathematics, Vol. 22, No. 1, pp. 203-217, 1970.

[2] H. Akaike, "A new look at statistical model identification", IEEE Trans. on Automatic Control, Vol. 19, No. 6, pp. 716-723, 1974.

[3] J. Aldrich, "R. A. Fisher and the making of maximum likelihood 1912-1922", Statistical Science, Vol. 12, pp. 162-176, 1997.

[4] R. Aringhieri, E. Damiani, S. De, C. D. Vimercati, S. Paraboschi, P. Samarati, "Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems", Journal of the American Society for Information Science and Technology, Vol. 57, No. 4, pp. 528-537, 2006.

[5] R. J. Aumann, "Game theory", Vol. 2, The New Palgrave: A Dictionary of Economics, 1987.

[6] F. Azzedin, A. Ridha, A. Rizvi, "Fuzzy trust for peer-to-peer based systems", Proc. world academy of science, engineering and technology, Vol. 21, pp. 123-127, 2007.

[7] M. Blaze, J. Feigenbaum, J. Lacy, "Decentralized Trust Management", Proc. IEEE Symposium on Security and Privacy, pp. 164-173, 1996.

[8] M. Blaze, J. Feigenbaum, J. Ioannidis, A. D. Keromytis, "The role of trust management in distributed systems security", Secure Internet Programming: Security Issues for Mobile and Distributed Objects, Vitek and Jensen (Eds), Springer-Verlag, 1999.

[9] M. Blaze, J. Feigenbaum, M. Strauss, "Compliance checking in the PolicyMaker trust management system", Lecture Notes in Computer Science, Vol. 1465, pp. 1439-1456, 1998.

[10] A. Boukerche, X. Li, "An agent-based trust and reputation management scheme for Wireless Sensor Networks", IEEE Global Telecommunications Conference, Vol. 3, pp. 1857-1862, 2005.

[11] G. Box, G. M. Jenkins, G. C. Reinsel, "Time Series Analysis: Forecasting and Control", 4th edn. Wiley, Chichester, 2008.

[12] J. E. Cavanaugh, "A large-Sample model selection criterion based on Kullback's symmetric divergence", Statistics & Probability Letters, Vol. 42, No. 4, pp. 333-343, 1999.

[13] B. Chang, S. Kuo, Y. Liang, D. Wang, "Markov Chain-based trust model for analyzing trust value in distributed multicasting mobile ad hoc networks", IEEE Asia-Pacific Services Computing Conference, pp. 156-161, 2008.

[14] D. A. Dickey, W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root", Journal of the American Statistical Association, Vol. 74, No. 388, pp. 427-431, 1979.

[15] L. Eschenauer, V. D. Gligor, J. Baras, "On trust establishment in mobile ad hoc networks", Proc. 10th Int'1 Security Protocols Workshop, Vol. 2845, pp. 47-66, 2002.

[16] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks", ISI Research Report, pp. 65-66, 1987.

[17] S. Ganeriwal, L. Bolzano, M. B. Srivastava, "Reputation based framework for high integrity sensor networks", ACM Trans. on Sensor Networks, Vol. 4, No. 3, 2008.

[18] J. Gong, J. Chen, H. Deng, J. Wang, "A trust model combining reputation and credential", WASE International Conference on Information Engineering, pp. 635-638, 2009.

[19] M. Hansen, B. Yu, "Model selection and the principle of minimum description length", Journal of the American Statistical Association, Vol. 96, No. 454, pp. 746-774, 2001.

[20] Z. Huang, Z. S. Chalabi, "Use of time-series analysis to model and forecast wind speed", Journal of Wind Engineering and Industrial Aerodynamics, Vol. 56, No. 2-3, pp. 311-322, 1995.

[21] K. Y. Huang, C. J. Jane, "A hybrid model for stock market forecasting and portfolio selection based on ARX, grey system and RS theories", Expert Systems with Applications, Vol. 36, No. 3, pp. 5387-5392, 2009.

[22] J. Jaramillo, R. Srikant, "DARWIN: distributed and adaptive reputation mechanism for wireless ad hoc networks", Proc. 13th Annual ACM International Conference on Mobile Computing and Networking, pp. 87-98, 2007.

[23] A. Jøsang, R. Ismail, "The Beta reputation system", Proc. 15th Bled Electronic Commerce Conference, pp. 324-337, 2002.

[24] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks", Conf. World Wide Web Conference Series, pp. 640-651, 2003.

[25] B. Kovacevic, Z. Durovic, "Robust recursive time series modeling using ARX models with optimal exogenous inputs", Proc. 13th Int'l Conference on Digital Signal Processing, Vol. 2, pp. 911-914, 1997.

[26] Q. Li, H. Fan, E. Karlsson, "A delta MYWE algorithm for parameter estimation of noisy AR processes", IEEE Trans. on Signal Processing, Vol. 44, No. 5, pp. 1300-1303, 2002.

[27] Z. Li, X. Li, A. Nayak, I. Stojmenovic, V. Narasimhan, "Autoregression Models for Trust Management in Wireless Ad Hoc Networks", Proc. IEEE Globecom, 2011, to appear.

[28] X. Long, J. Joshi, "BaRMS: A Bayesian Reputation Management Approach for P2P systems", IEEE International Conference on Information Reuse and Integration, pp. 147-152, 2010.

[29] M. Momani, K. Aboura, S. Challa, "RBATMWSN: Recursive Bayesian Approach to Trust Management in Wireless Sensor Networks", 3rd International Conference on Intelligent Sensors, Sensor Networks and Information, pp. 347-352 , 2008.

[30] D. C. Montgomery, G. C. Runger, N. F. Hubele, "Engineering Statistics", John Wiley and Sons Inc., 2004.

[31] J. V. Neumann, O. Morgenstern, "Theory of games And economic behavior", Princeton University Press, 1944.

[32] T. Ogawa, H. Sonoda, S. Ishiwa, Y. Shigeta, "Distribution of the estimators for autoregressive time series with a unit root", Brain Topography, Vol. 6, No. 1, pp. 3-11, 1993.

[33] G. Schwarz, "Estimating the dimension of a model", Annals of Statistics, Vol. 6, No. 2, pp. 461-464, 1978.

[34] A. A. Selcuk, E. Uzun, M. R. Pariente, "A reputation-based trust management system for P2P networks", IEEE International Symposium on Cluster Computing and the Grid, pp. 251-258, 2004.

[35] M. Seredynski, P. Bouvry, M. A. Klopotek, "Modelling the evolution of cooperative behavior in ad hoc networks using a game based model", Proc. IEEE Symposium on Computational Intelligence and Games, pp. 96-103, 2007.

[36] X. Shen, H. Yu, J. Buford, "Handbook of peer-to-peer networking", Springer US, 2010.

[37] A. Singh, L. Liu, "TrustMe:anonymous management of trust relationships in decentralized P2P systems", 3rd International Conference on Peer-to-Peer Computing, pp. 142-149, 2003.

[38] P. Stoica, B. Friedlander, T. Soderstrom, "Least-squares, Yule-Walker, and overdetermined Yule-Walker estimation of AR parameters: a Monte Carlo analys is of finite-sample properties", International Journal of Control, Vol. 43, No. 1, pp. 13-27, 1986.

[39] G. Theodorakopoulos, J. S. Baras, "On trust models and trust evaluation metrics for ad hoc networks", IEEE Journal on Selected Areas in Communications, Vol. 24, No. 2, pp. 318-328, 2006.

[40] M. Toorani, A. B. Shirazi, "LPKI - a Lightweight Public Key Infrastructure for the mobile environments", Proc. 11th IEEE International Conference on Communication Systems, pp. 162-166, 2008.

[41] T. Wölfl, "Public-Key-Infrastructure Based on a Peer-to-Peer Network", Proc. 38th Annual Hawaii International Conference on System Sciences, Vol. 7, pp. 200-204, 2005.

[42] W. Xu, F. Liu, F. Li, H. Wu, X. Jin, "Parameter estimations in linear regression models with AR(2) errors in which the parameters have a special relationship", Proc. Int'l Conference on Computational Intelligence and Software Engineering, pp. 1-4, 2009.

[43] L. Xu, Y. Zhang, "A New reputation-based trsut management strategy for clustered ad hoc networks", International Conference on Networks Security, Wireless Communications and Trusted Computing, pp. 116-119, 2009.

[44] K. Zeng, "Pseudonymous PKI for ubiquitous computing", Lecture Notes in Computer Science, Vol. 4043, pp. 207-222, 2006.

[45] J. Zhang, "Trust management based on fuzzy sets theory for P2P networks", WRI World Congress on Software Engineering, pp. 461-465, 2009.

[46] Wikipedia, the free encyclopedia.

[47] Merriam-Webster dictionary, http://www.merriam-webster.com/dictionary-tb